

Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences

KARL D.D. WILLIS, Autodesk Research, USA

YEWEN PU, Autodesk Research, USA

JIELIANG LUO, Autodesk Research, USA

HANG CHU, Autodesk Research, Canada

TAO DU, Massachusetts Institute of Technology, USA

JOSEPH G. LAMBOURNE, Autodesk Research, United Kingdom

ARMANDO SOLAR-LEZAMA, Massachusetts Institute of Technology, USA

WOJCIECH MATUSIK, Massachusetts Institute of Technology, USA

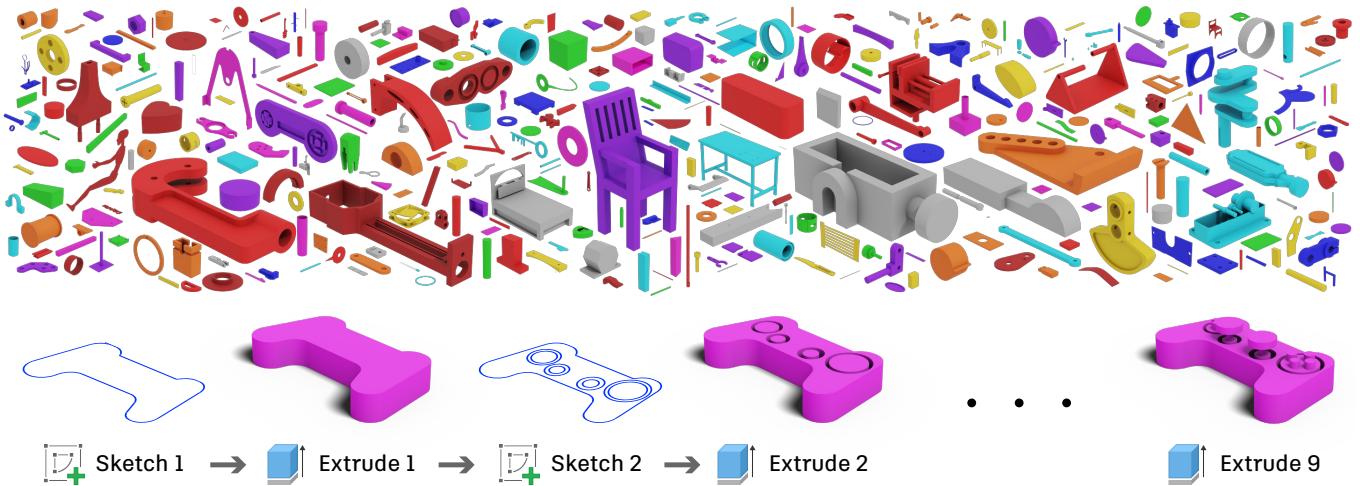


Fig. 1. Top: A subset of designs containing ground-truth CAD programs represented as construction sequences from the *Fusion 360 Gallery* reconstruction dataset. Bottom: An example construction sequence using the *sketch* and *extrude* modeling operations with built-in Boolean operations.

Parametric computer-aided design (CAD) is a standard paradigm used to design manufactured objects, where a 3D shape is represented as a program supported by the CAD software. Despite the pervasiveness of parametric CAD and a growing interest from the research community, currently there does not exist a dataset of realistic CAD models in a concise programmatic form. In this paper we present the *Fusion 360 Gallery*, consisting of a simple

Authors' addresses: Karl D.D. Willis, Autodesk Research, San Francisco, California, USA, karl.willis@autodesk.com; Yewen Pu, Autodesk Research, San Francisco, California, USA; Jieliang Luo, Autodesk Research, San Francisco, California, USA; Hang Chu, Autodesk Research, Toronto, Ontario, Canada; Tao Du, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA; Joseph G. Lambourne, Autodesk Research, Soho, London, United Kingdom; Armando Solar-Lezama, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA; Wojciech Matusik, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM. 0730-0301/2021/8-ART54 \$15.00
https://doi.org/10.1145/3450626.3459818

language with just the *sketch* and *extrude* modeling operations, and a dataset of 8,625 human design sequences expressed in this language. We also present an interactive environment called the *Fusion 360 Gym*, which exposes the sequential construction of a CAD program as a Markov decision process, making it amenable to machine learning approaches. As a use case for our dataset and environment, we define the CAD reconstruction task of recovering a CAD program from a target geometry. We report results of applying state-of-the-art methods of program synthesis with neurally guided search on this task.

CCS Concepts: • Computing methodologies → Parametric curve and surface models.

Additional Key Words and Phrases: Computer aided design, CAD, dataset, construction, geometry synthesis, reconstruction

ACM Reference Format:

Karl D.D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2021. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences. *ACM Trans. Graph.* 40, 4, Article 54 (August 2021), 24 pages. https://doi.org/10.1145/3450626.3459818

1 INTRODUCTION

The manufactured objects that surround us in everyday life are represented programmatically in computer-aided design (CAD) software as a sequence of 2D and 3D modeling operations. Parametric CAD files contain programmatic information that is critical for documenting design intent, maintaining editability, and compatibility with downstream simulation and manufacturing. Embedded within these designs is the knowledge of domain experts who precisely define a sequence of modeling operations to form 3D shapes. We believe having access to a real-world collection of human design sequences, and the ability to execute them, is critical for future advances in CAD that leverage learning-based approaches.

Learning-based approaches show great potential, both for solving existing problems such as reverse engineering [Buonomici et al. 2018], and for providing entirely new kinds of functionality which would be unimaginable using traditional techniques. Recent advances in neural networks have spurred new interest in data driven approaches to generating CAD programs, tackling both the forward problem of 3D shape generation [Jones et al. 2020; Li et al. 2020b; Mo et al. 2019a] and the inverse problem of recovering CAD programs from a target geometry [Ellis et al. 2019; Kania et al. 2020; Sharma et al. 2017; Tian et al. 2019]. However, progress has been inhibited by the lack of a human designed dataset of ground-truth CAD programs, written in a simple yet expressive Domain Specific Language (DSL) and an environment to execute them.

We take a step towards this goal by introducing the first dataset of human designed CAD geometries, paired with their ground-truth CAD programs represented as construction sequences, along with a supporting execution environment to make learning-based approaches amendable to real CAD construction tasks. Our dataset contains 8,625 CAD programs represented entirely in a simple language allowing sketches to be created and then extruded. With just the *sketch* and *extrude* modeling operations, that also incorporate Boolean operations, a highly expressive range of 3D designs can be created (Figure 1). We provide an interactive environment called the *Fusion 360 Gym*, which can interpret the language of sketch and extrude, providing a geometric data structure as feedback after each operation, simulating the iterative construction process of a human designer.

As a use case for our dataset and environment, we standardize the problem of programmatic CAD reconstruction from a target geometry using a learning-based approach. We provide a benchmark, consisting of a training set of 6,900 designs and a test set of 1,725 designs, and a set of evaluation criteria. We then develop neurally guided search approaches for the CAD reconstruction task on this benchmark. Our algorithm consists of first training a policy, a message passing network (MPN) with a novel encoding of state and action, using imitation learning on ground truth construction sequences. At inference time the algorithm employs search, leveraging the learned neural policy to repeatedly interact with the *Fusion 360 Gym* environment until a correct CAD program is discovered. This approach is able to recover a correct CAD program for 67.5% of designs in the test set with a budget of 100 interactions between the agent and the *Fusion 360 Gym*, averaging < 20 sec solve time per design. This paper makes the following contributions:

- We present the *Fusion 360 Gallery* reconstruction dataset, containing 8,625 human designed CAD programs, expressed in a simple yet expressive language of *sketch* and *extrude*.
- We introduce an environment called the *Fusion 360 Gym*, capable of executing the language of *sketch* and *extrude* and providing a geometric data structure as feedback after each operation.
- We standardize the task of CAD reconstruction from input geometry and use a learning-based approach with neurally guided search to produce results on real world data for the first time.

2 RELATED WORK

CAD Datasets. Existing 3D CAD datasets have largely focused on providing mesh geometry [Chang et al. 2015; Kim et al. 2020; Mo et al. 2019b; Wu et al. 2015; Zhou and Jacobson 2016]. However, the de facto standard for parametric CAD is the boundary representation (B-Rep) format, containing valuable analytic representations of surfaces and curves suitable for high level control of 3D shapes. B-Reps are collections of trimmed parametric surfaces along with topological information which describes adjacency relationships between them [Weiler 1986]. B-Rep datasets have recently been made available with both human designed [Koch et al. 2019] and synthetic data [Jayaraman et al. 2020; Starly 2020; Zhang et al. 2018]. Missing from these datasets is programmatic construction sequence information containing the knowledge of how each shape is defined and created. Although the ABC dataset includes some additional construction information in a proprietary format provided by the Onshape CAD software, missing information can only be retrieved by querying the OnShape API. Combined with sparse documentation, this makes it difficult to interpret the construction information. We are unaware of any method that can be used to rebuild designs in the ABC dataset from the provided construction information, a key requirement for tasks related to CAD construction. We believe it is critical to understand not only *what* is designed, but *how* that design came about.

Parametric CAD programs contain valuable information on the construction history of a design. Schulz et al. [2014] provide a standard collection of human designs with full parametric history, albeit a limited set of 67 designs in a proprietary format. SketchGraphs [Seff et al. 2020] narrows the broad area of parametric CAD by focusing on the underlying 2D engineering sketches, including sketch construction sequences. Freehand 2D sketch datasets also tackle the challenge of understanding design by looking at the sequence of user actions [Eitz et al. 2012; Gryaditskaya et al. 2019; Sangkloy et al. 2016]. In the absence of human designed sequential 3D data, learning-based approaches have instead leveraged synthetic CAD construction sequences [Ellis et al. 2019; Li et al. 2020b; Sharma et al. 2017; Tian et al. 2019]. The dataset presented in this paper is the first to provide human designed 3D CAD construction sequence information suitable for use with machine learning. Table 1 provides a feature comparison of related CAD datasets.

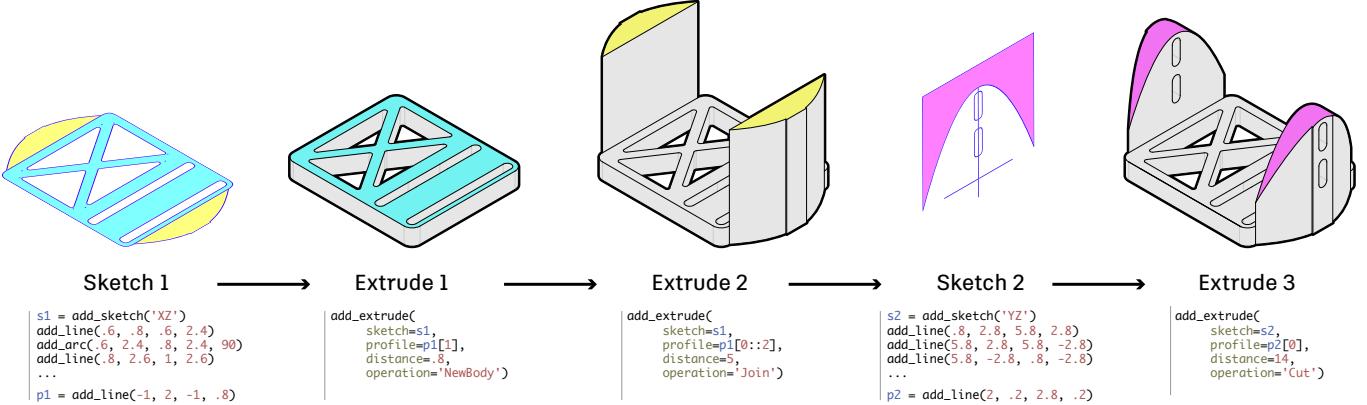


Fig. 2. An example design sequence from the dataset with associated CAD program. Sketch elements form profiles that are sequentially extruded to *join* (Extrude 1, Extrude 2) or *cut* (Extrude 3) geometry using Boolean operations. The colored areas show the sketch profiles that partake in each extrusion.

Table 1. Comparison of related CAD datasets. For each dataset, we report the number of designs (#), the design representation (**B-Rep**, **Mesh**, or **Sketch**), whether it includes a construction sequence capable of rebuilding the final design (**Seq.**), and whether it contains human annotated labels for tasks such as shape classification (**Label**). The *F360 Gallery* row indicates our dataset.

Dataset	#	B-Rep	Mesh	Sketch	Seq.	Label
ShapeNet	3M+			✓		✓
ABC	1M+	✓	✓			
Thingi10k	10,000		✓			✓
SketchGraphs	15M+			✓	✓	
<i>F360 Gallery</i>	8,625	✓	✓	✓	✓	

3D Shape Generation. The forward problem of 3D shape generation has been explored extensively in recent years using learning-based approaches. Neural network based generative models are often used to enable previously challenging functionality such as shape interpolation and synthesis. Notable approaches to this problem include leveraging knowledge of object structure [Gao et al. 2019; Li et al. 2020a; Mo et al. 2019a; Schor et al. 2019] or learning from a sequence of events to generate 3D shapes [Jones et al. 2020; Li et al. 2020b; Nash et al. 2020; Sung et al. 2017; Wu et al. 2020; Zou et al. 2017]. Unique to our work is the challenge of learning from real sequential human design data, requiring a state and action representation suitable for the language of *sketch* and *extrude*.

CAD Reconstruction. The inverse task of CAD reconstruction involves recovering a CAD program, represented as a sequence of modeling operations, from input such as B-Reps, triangle meshes, or point clouds. Despite extensive prior work [Shah et al. 2001], CAD reconstruction remains a challenging problem as it requires deductions on both continuous parameters (e.g., extracting the dimensions of primitives) and discrete operations (e.g., choosing a proper operation for the next step), leading to a mixed combinatorial search space. To recover the sequence of operations, traditional methods typically run global search methods (e.g., evolutionary algorithms as in Hamza and Saitou [2004], Weiss [2009], Friedrich et al. [2019],

and Fayolle and Pasko [2016]) with heuristic rules to prune the search space [Buchele 2000; Buchele and Crawford 2003; Buchele and Roles 2001; Shapiro and Vossler 1993]. Heuristic approaches are also available in a number of commercial software tools, often as a user-guided semi-automatic system [Autodesk 2012; Dassault 2019] to aid with file conversion between CAD systems. These traditional algorithms operate by removing faces from the B-rep body and reapplying them as parametric modeling operations. This strategy can recover the later modeling operations, but fail to completely rebuild the construction sequence from the first step. We instead tackle the task of recovering the entire construction sequence from the first extrusion. Another approach is using program synthesis [Du et al. 2018; Nandi et al. 2017, 2018, 2020] to infer CAD programs written in DSLs from given shapes. CAD reconstruction is also related to the inverse procedural modeling problem [Stava et al. 2014; Talton et al. 2011; Vanegas et al. 2012], which attempts to reverse-engineer procedures that can faithfully match a given target.

Compared to the rule-based or grammar-based methods above, learning-based approaches can potentially learn the rules that are typically hard-coded, automate scenarios that require user-input, and generalize when confronted with unfamiliar geometry. One early work is CSGNet [Sharma et al. 2017], which trains a neural network to infer the sequence of Constructive Solid Geometry (CSG) operations based on visual input. More recent works along this line of research include [Chen et al. 2020; Ellis et al. 2019; Kania et al. 2020; Tian et al. 2019]. Typically associated with these methods are a customized DSL, such as CSG, that parameterizes the space of geometry, some heuristic rules that limit the search space, and a neural network generative model. Lin et al. [2020] reconstruct input shapes with a dual action representation that first positions cuboids and then edits edge-loops for refinement. Although editing edge-loops of cuboids may be a suitable modeling operation in artistic design, it is not as expressive or precise as the sketch and extrude operations used in real mechanical components. Additionally, Lin et al. [2020] choose to train and evaluate their network on synthetic data due to the lack of a benchmark dataset of CAD construction sequences, a space that our work aims to fill. Our approach is the first to apply a



Fig. 3. Modeling operations other than *sketch* and *extrude* are suppressed to expand the data quantity. An example design before (left) and after (right) the fillet modeling operation is suppressed.

learning-based method to reconstruction using common *sketch* and *extrude* CAD modeling operations from real human designs.

3 FUSION 360 GALLERY DSL AND RECONSTRUCTION DATASET

The *Fusion 360 Gallery* reconstruction dataset consists of 8,625 designs produced by users of the CAD software Autodesk Fusion 360 and submitted to the publicly available Autodesk Online Gallery [Autodesk 2015]. The data and supporting code is publicly available via GitHub¹ with a license allowing non-commercial research similar to the ImageNet [Deng et al. 2009] license. We created the dataset from approximately 20,000 designs in the native Fusion 360 CAD file format. We focus on the *sketch* and *extrude* modeling operations for two main reasons: 1) *sketch* and *extrude* are the two most common CAD modeling operations used in 84% and 79% of designs in the original dataset respectively; >3x more common than operations such as fillet and chamfer, and 2) we seek to balance design expressivity with a tractable problem for learning-based approaches; restricting the modeling operations to *sketch* and *extrude* greatly simplifies the descriptive complexity compared to the full range of CAD modeling operations. We generate the as-designed sequence of *sketch* and *extrude* modeling operations by parsing the parametric history of the Fusion 360 CAD files. Multi-component assemblies are divided into separate designs representing the constituent parts, e.g. the blade of a pocket knife. Modeling operations other than *sketch* and *extrude* are suppressed to expand the data quantity. Figure 3 shows an example of suppressing a fillet operation, allowing the resulting design to be included in the dataset. Figure 4 shows a random sampling of the designs in the dataset grouped by the number of extrude operations used.

Each design is represented as a program expressed in a DSL, forming a simplified wrapper around the underlying Fusion 360 Python API [Autodesk 2014]. Each design consists of a sequence

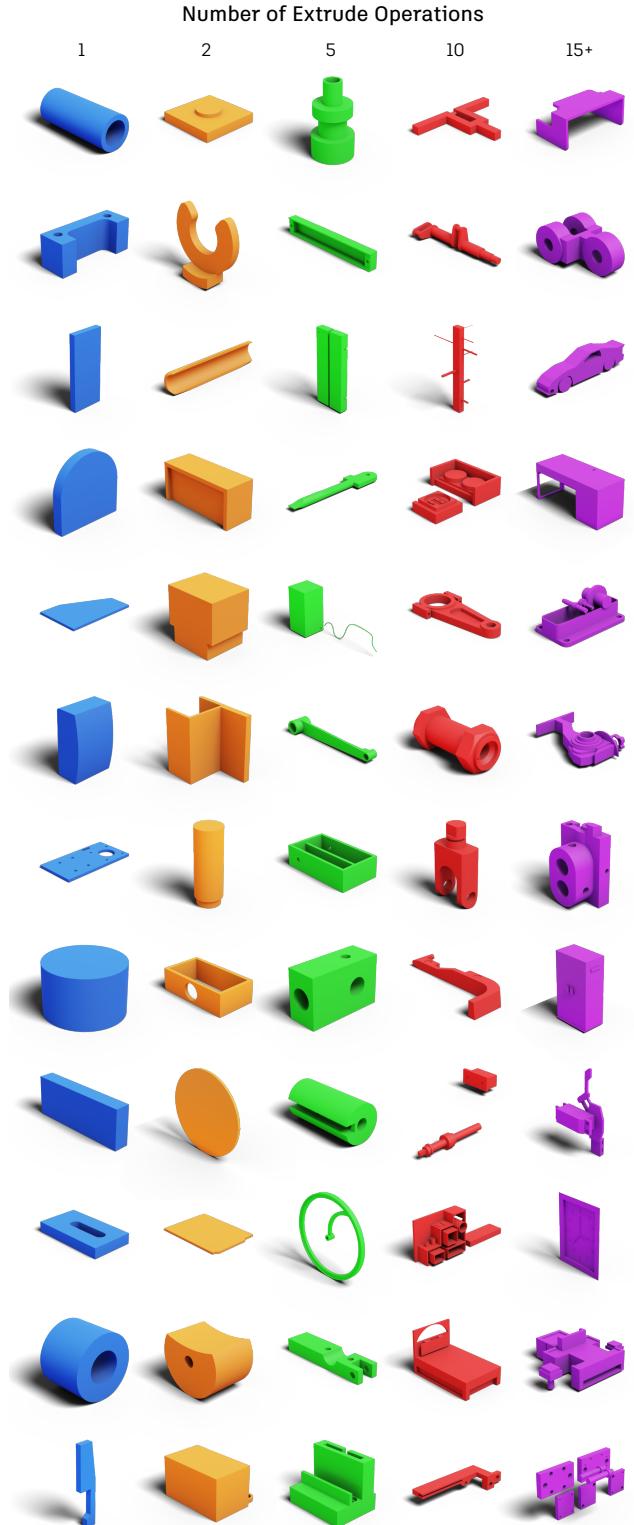


Fig. 4. A random sampling of designs from the Fusion 360 Gallery reconstruction dataset, grouped by the number of extrude operations.

¹Dataset website: <https://github.com/AutodeskAILab/Fusion360GalleryDataset>

Table 2. The grammar for the *Fusion 360 Gallery* domain-specific language. A program consists of a sequence of *sketch* and *extrude* operations that iteratively modify the current geometry.

P	$::= G; [X]$
X	$::= S \mid E$
S	$::= \text{add_sketch}(I); [D]$
D	$::= L \mid A \mid C$
L	$::= \text{add_line}(N, N, N, N)$
A	$::= \text{add_arc}(N, N, N, N, N)$
C	$::= \text{add_circle}(N, N, N)$
E	$::= \text{add_extrude}([I], N, O)$
I	$::= \text{identifier}$
N	$::= \text{number}$
O	$::= \text{new body} \mid \text{join} \mid \text{cut} \mid \text{intersect}$

of *sketch* and *extrude* operations that iteratively modifies the current geometry (Figure 2). We specify the core language here, and provide information on additional constructs, such as sketching of splines and double-sided extrudes, in Section A.1 of the appendix. The *Fusion 360 Gallery* DSL is a *stateful* language consisting of a single global variable G , representing the current geometry under construction, and a sequence of commands $[X]$ that iteratively modifies the current geometry G . Each command can be either a *sketch* S or an *extrude* E operation. A grammar describing the core DSL is shown in Table 2.

3.1 Current Geometry

The current geometry G is the single *global state* that is updated with the sequence of commands $[X]$. It is a data structure representing all geometric information that would be available to a designer in the construction process using Fusion 360: such as inspecting different aspects of the geometry, and referencing its components for further modifications.

Boundary Representation. B-Rep is the primary geometry format provided in the dataset and the native format in which designs were created, making it a natural representation for the current geometry. G represents a collection of sketch or B-Rep entities, which can be referenced from the construction sequence through identifier I . B-Rep bodies can be expressed as a face adjacency graph, as later described in Section 4.1.

Execution. Crucially, the current geometry G is iteratively updated through the sequence of commands $[X]$. After each command X , the interpreter uses the underlying Fusion 360 Python API to generate an updated geometry. After all the commands $[X]$ are executed, we obtain the final geometry, G_t .

Storage. In addition to the program P , *Fusion 360 Gym* stores the final geometry G_t as a .smt file, the native B-Rep format used by Fusion 360, and neutral .step files that can be used with other CAD systems. B-Rep entities, such as bodies and faces, can be referenced from the construction sequence back to entities in the .smt file. A mesh representation of G_t is stored in .obj format representing a triangulated version of the B-Rep. Each B-Rep face is labeled as a

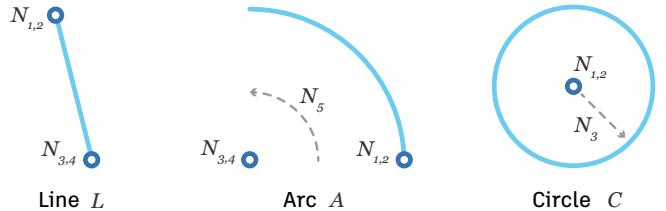


Fig. 5. Sketch commands used to create a Line L , Arc A , and Circle C .

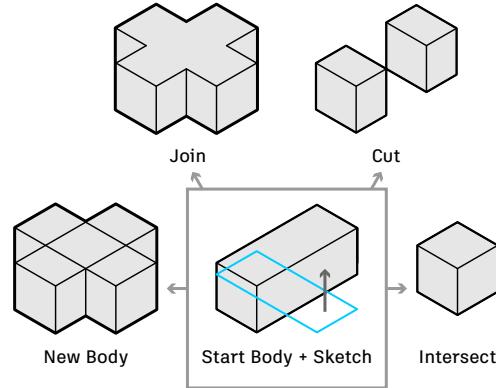


Fig. 6. Extrude operations include the ability to Boolean with other geometry. From the start body shown in the center, a sketch is extruded to form a new body overlapping the start body, join with the start body, cut out of the start body, or intersect with the start body.

group of triangles in the .obj file with the B-Rep face identifier as the group name. This allows the triangles to be traced back to the B-Rep face and associated extrude operation. Any intermediate geometry G can also be exported in these file formats with the API.

3.2 Sketch

A sketch operation, S , is stated by specifying the plane on which the sketch will be created using the *add_sketch*(I) command. I is a plane identifier, which allows for identification of the three canonical planes XY, YZ, XZ along with other planar faces present in the current geometry G . Following the identification of a sketch plane, one can add a sequence of sketch commands $[D]$, where each command is either a line L , arc A , or circle C (Figure 5). Line, arc, and circle represent 95% of curves in the dataset. A line command L is specified by four numbers, representing the coordinates for the start and end points. A circle command C is specified by three numbers, two representing the circle's center and one representing its radius. An arc command A is specified by five numbers, representing the start point, the arc's center point, and the angle which the arc subtends. The coordinates for the line L , arc A , and circle C are specified with respect to the coordinate system of the chosen sketch plane I in G . Executing a sketch S command creates a list of new profiles in the current geometry G , consisting of enclosed regions resulting from the sketch.

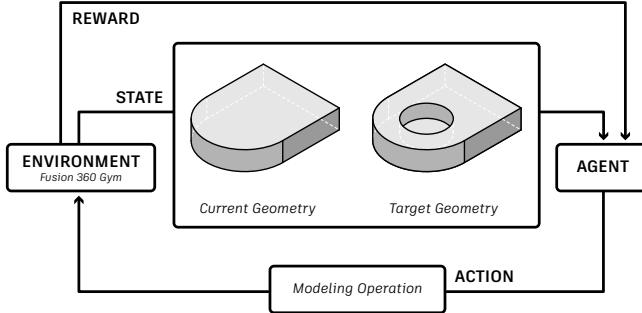


Fig. 7. The *Fusion 360 Gym* environment interacts with an *agent* in a sequential decision making scenario. The *state* contains the current and target geometries. The agent outputs an action, in the form of a modeling operation, that advances the current geometry towards the target.

3.3 Extrude

An extrude operation E takes a list of identifiers, $[I]$, referencing a list of profiles in the current geometry G , and extrudes them from 2D into 3D. A signed distance parameter N defines how far the profile is extruded along the normal direction. The Boolean operation O specifies whether the extruded 3D volume is added to, subtracted from, or intersected with other 3D bodies in the design. Figure 6 shows a start body and sketch (center) that is extruded to form two separate overlapping bodies, joined to form a single body, cut through the start body to split it in two, or intersected with the start body. Additional extrude options are available such as two-sided extrude, symmetrical extrude, and tapered extrude (See Section A.1.6 of the appendix). Executing an extrude operation E results in an updated list of bodies in the current geometry G . The combination of expressive sketches and extrude operations with built in Boolean capability enables a wide variety of designs to be constructed from only two modeling operations (Figure 1).

4 FUSION 360 GYM

Together with the dataset we provide an open source environment, called the *Fusion 360 Gym*, for standardizing the CAD reconstruction task for learning-based approaches. The *Fusion 360 Gym* further simplifies the *Fusion 360 Gallery* DSL and serves as the environment that interacts with an intelligent agent for the task of CAD reconstruction (Figure 7). Just as a designer can iteratively interact with a CAD software system in a step-by-step fashion, comparing at each step the target geometry to be recovered and the current geometry they have created so-far, the *Fusion 360 Gym* provides the intelligent agent with the same kind of interaction. Specifically, the *Fusion 360 Gym* formalizes the following Markov Decision Process:

- **state:** Contains the current geometry, and optionally, the target geometry to be reconstructed. We use a B-Rep face-adjacency graph as our state representation.
- **action:** A modeling operation that allows the agent to modify the current geometry. We consider two action representations: sketch extrusion and face extrusion.

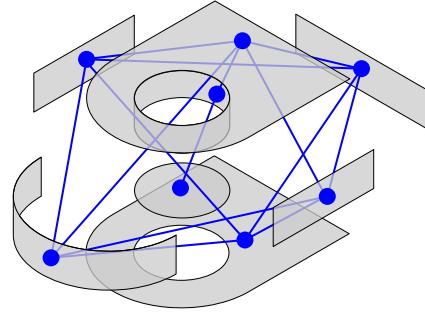


Fig. 8. For state representation we use a face adjacency graph with B-Rep faces as graph vertices and B-Rep edges as graph edges.

- **transition:** *Fusion 360 Gym* implements the transition function that applies the modeling operation to update the current geometry.
- **reward:** The user can define custom reward functions depending on the task. For instance, the agent might receive a reward of 1 if the current geometry exactly matches the target geometry.

4.1 State Representation

In order for an agent to successfully reconstruct the target geometry, it is important that we have a suitable state representation. In the *Fusion 360 Gym*, we use a similar encoding scheme to Jayaraman et al. [2020] and represent the current and target geometry with a B-Rep face-adjacency graph [Ansaldi et al. 1985], which contains additional information amenable to a learning agent not present in the language of the *Fusion 360 Gallery* DSL (Figure 8). Crucial to this encoding are the *geometric* features of the elements, such as point-locations, and *topological* features specifying how these elements are connected to each other. Specifically, the vertices of the face-adjacency graph represent B-Rep faces (trimmed parametric surfaces) in the design, with graph vertex features representing the size, orientation, and curvature of the faces. The edges of the face-adjacency graph represent B-Rep edges in the design, that connect the adjacent B-Rep faces to each other. Additional details are provided in Section A.3.2 of the appendix.

4.2 Action Representation

In the *Fusion 360 Gym* we support two action representations encompassing different modeling operations: *sketch extrusion* and *face extrusion*.

- 4.2.1 *Sketch Extrusion.* Sketch extrusion mirrors the *Fusion 360 Gallery* DSL closely. In this scheme, the agent must first select a sketch plane, draw on this plane using a sequence of curve primitives, such as lines and arcs, to form closed loop profiles. The agent then selects a profile to extrude a given distance and direction (Figure 9, top). Using this representation it is possible to construct novel geometries by generating the underlying sketch primitives and extruding them by an arbitrary amount. Although all designs in the *Fusion 360 Gallery* reconstruction dataset can be constructed using sketch extrusion, in practice this is challenging. Benko et al. [2002]

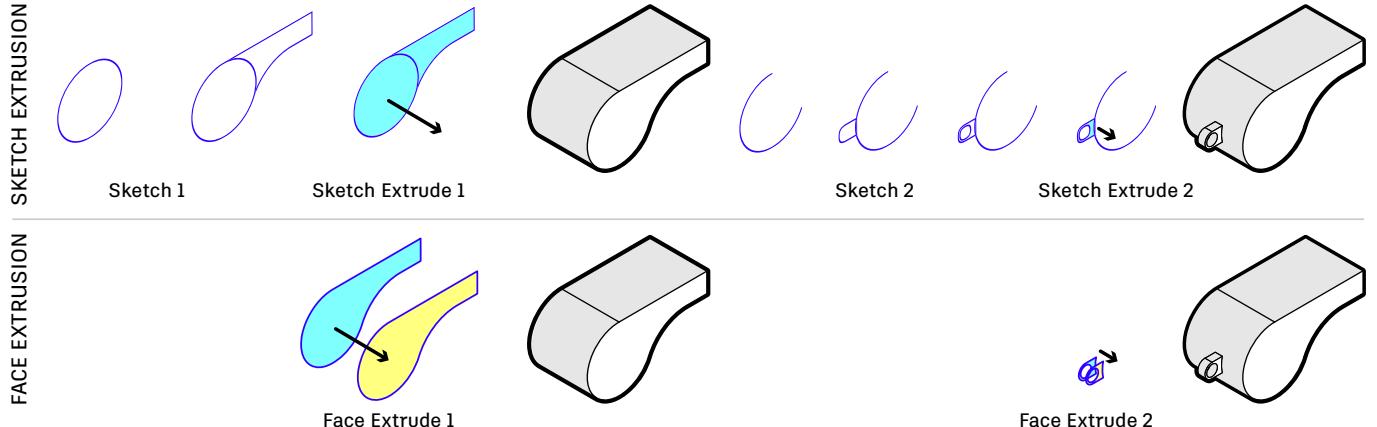


Fig. 9. Action representations supported by the *Fusion 360 Gym* include low-level sketch extrusion (top) and simplified face extrusion (bottom).

show that to generate sketches suitable for mechanical engineering parts, the curve primitives often need to be constructed alongside a set of constraints which enforce regularities and symmetries in the design. Although the construction of constraint graphs is feasible using techniques like the one shown by Liao et al. [2019], enforcing the constraints requires a complex interaction between the machine learning algorithm and a suitable geometric constraint solver, greatly increasing the algorithm complexity. We alleviate this problem by introducing a simplified action representation, called *face extrusion*, that is well suited to learning-based approaches.

4.2.2 Face Extrusion. In face extrusion, a face from the target design is used as the extrusion profile rather than a sketch profile (Figure 9, bottom). This is possible because the target design is known in advance during reconstruction. An action a in this scheme is a triple $\{face_{start}, face_{end}, op\}$ where the start and end faces are parallel faces referenced from the target geometry, and the operation type is one of the following: *new body*, *join*, *cut*, *intersect*. The start face defines the extrusion profile and the end face defines the distance to be extruded and does not need to match the shape of the start face. Target constrained reconstruction using face extrusion has the benefit of narrowly scoping the prediction problem with shorter action sequences and simpler actions. Conversely, not all geometries can be reconstructed with this simplified strategy due to insufficient information in the target, e.g., Extrude 3 in Figure 2 cuts across the entire design without leaving a start or end face.

4.3 Synthetic Data Generation

The *Fusion 360 Gym* supports generation of synthetic designs for data augmentation. In addition to procedurally generated synthetic data, semi-synthetic data can be generated by taking existing designs and modifying or recombining them. For instance, we can randomly perturb the sketches and the extrusion distances, and even ‘graft’ sketches from one design onto another. We also support distribution matching of parameters, such as the number of faces, to ensure that synthetic designs match a human designed dataset distribution. Learning-based systems can leverage semi-synthetic data to expand the number of samples in the *Fusion 360 Gallery* reconstruction

dataset. In Section 6.2 we evaluate the performance of synthetic and semi-synthetic data for the CAD reconstruction task. We provide examples of synthetic data in Figure 15 and commands for the *Fusion 360 Gym* in Section A.2 of the appendix.

5 CAD RECONSTRUCTION TASK

5.1 Task Definition

The goal of CAD reconstruction is to recover a program, represented as a sequence of modeling operations used to construct a CAD model with only the geometry as input. This task can be specified using different input geometry representations, including B-Rep, mesh, or point cloud, with progressively lower fidelity. Each representation presents a realistic scenario where parametric CAD information is absent and needs to be recovered. Given a target geometry G_t , we wish to find a sequence of CAD modeling operations (actions) $\mathcal{A} = \{a_0, a_1, \dots\}$ such that, once executed in a CAD software system, results in a geometry G where every point in space is in its interior, if and only if, it is also in the interior of G_t .

5.2 Evaluation Metrics

We prescribe three evaluation metrics, IoU, exact reconstruction, and conciseness. IoU measures the intersection over union of G and G_t : $\text{iou}(G, G_t) = |G \cap G_t| / |G \cup G_t|$. Exact reconstruction measures whether $\text{iou}(G, G_t) = 1$. As multiple correct sequences of CAD modeling operations exist, a proposed reconstruction sequence \mathcal{A} need not match the ground truth sequence $\hat{\mathcal{A}}_t$ provided an exact reconstruction is found. To measure the quality of exact reconstructions we consider the conciseness of the construction sequence. Let $\text{concreteness}(\mathcal{A}, \hat{\mathcal{A}}_t) = |\mathcal{A}| / |\hat{\mathcal{A}}_t|$, where a score ≤ 1 indicates the agent found an exact reconstruction with equal or fewer steps than the ground truth, and a score > 1 indicates more inefficient exact reconstructions.

5.3 Neurally Guided Search

We now present a method for CAD reconstruction using neurally-guided search [Devlin et al. 2017; Ellis et al. 2019; Kalyan et al. 2018; Tang et al. 2019] from *B-Rep* input using *face extrusion* modeling

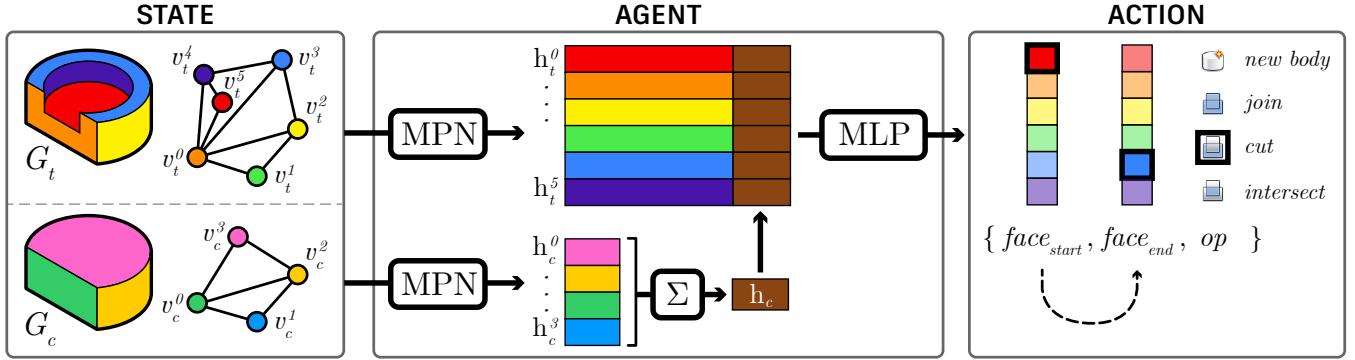


Fig. 10. Given a *state* comprising the target geometry G_t and current geometry G_c , both of which are represented as a graph, the *agent* uses message passing networks (MPNs) to predict an *action* as a face extrusion modeling operation. The first MPN in the bottom branch produces a set of node embedding vectors $\mathbf{h}_c^0 \dots \mathbf{h}_c^3$, which are summed over to produce the hidden vector for the current geometry \mathbf{h}_c . Another MPN in the top branch produces a set of node embedding vectors $\mathbf{h}_t^0 \dots \mathbf{h}_t^5$, which are concatenated with \mathbf{h}_c to predict the *action*. We condition the end face prediction on the predicted start face. Colors in the figure correspond to different graph nodes.

operations. The training phase consists of imitation learning, where a policy is trained to imitate a known construction sequence from a given geometry. The testing / inference phase leverages search, where the search algorithm repeatedly samples the trained policy for actions and applies these actions in the environment to generate a set of candidate reconstruction sequences.

5.3.1 Imitation Learning. To perform imitation learning, we leverage the fact that we have the ground truth sequence of modeling operations (actions) $\hat{\mathcal{A}}_t = \{\hat{a}_{t,0} \dots \hat{a}_{t,n-1}\}$ for each design G_t in the dataset. We feed the ground truth action sequence $\hat{\mathcal{A}}_t$ into the *Fusion 360 Gym*, starting from the empty geometry G_0 , and output a sequence of partial constructions $G_{t,1} \dots G_{t,n}$ where $G_{t,n} = G_t$. We then collect the supervised dataset $\mathcal{D} = \{(G_0, G_t) \rightarrow \hat{a}_{t,0}, (G_{t,1}, G_t) \rightarrow \hat{a}_{t,1} \dots\}$ and train a supervised agent π_θ that takes the pair of current-target constructions (G_c, G_t) to a modeling operation action a_c , which would transform the current geometry closer to the target. Formally, we optimize the expected log-likelihood of correct actions under the data distribution:

$$E_{(G_c, G_t) \sim \mathcal{D}} \left[\log \pi_\theta(a_c | (G_c, G_t)) \right] \quad (1)$$

5.3.2 Agent. The agent (Figure 10) takes a pair of geometries (G_c, G_t) as state, and outputs the corresponding face-extrusion action $a = \{face_{start}, face_{end}, op\}$. The two geometries G_c, G_t are given using a face-adjacency graph similar to Jayaraman et al. [2020], where the graph vertexes represent the faces of the geometry, with vertex features calculated from each face: 10×10 grid of 3D points, normals, and trimming mask, in addition to the face surface type. The 3D points are global xyz values sampled in UV parameter space of the face. The edges define connectivity of adjacent faces. Inputs are encoded using two *separate* message passing networks [Gilmer et al. 2017; Kipf et al. 2018; Kipf and Welling 2016] aggregating messages along the edges of the graph. The encoded vectors representing the *current* geometry are summed together (\mathbf{h}_c in Figure 10), and concatenated with the encoded vertexes of the target geometry ($\mathbf{h}_t^0 \dots \mathbf{h}_t^5$ in Figure 10). The concatenated vectors are used to output

the action using a multi-layer perceptron (MLP), with the end face conditioned on the vertex embedding of the predicted start face.

We denote the learned vertex embedding vectors produced by the two MPN branches as $\{\mathbf{h}_c^i\}$ and $\{\mathbf{h}_t^j\}$ for the current output and target graphs, respectively. We estimate the probability of the k -th operation type, and the j -th face being the start face or end face as:

$$P_{op}^k = F_{op}(\mathbf{h}_c), \quad \mathbf{h}_c = \sum_i \mathbf{h}_c^i \quad (2)$$

$$P_{start}^j = \text{softmax}_j(F_{start}(\mathbf{h}_t^j, \mathbf{h}_c)) \quad (3)$$

$$P_{end}^j = \text{softmax}_j(F_{end}(\mathbf{h}_t^j, \mathbf{h}_c)), \quad s.t. \quad j = \arg\max_j P_{start}^j \quad (4)$$

where F_{op} , F_{start} , and F_{end} denote linear layers that take the concatenated vectors as input.

5.3.3 Search. Given a neural agent $\pi_\theta(a | (G_c, G_t))$ capable of furthering a current geometry toward the target geometry, we can amplify its performance at test time using search. This allows us to explore multiple different reconstruction sequences at once, at the expense of extended interactions with the environment. By leveraging search, one gets the benefit of scaling: the larger budget we have to interact with the environment, the more likely we are going to succeed in recovering a working reconstruction sequence. The effectiveness of search is measured against a *search budget*, which in our case, is the number of *environment steps* executed in the *Fusion 360 Gym*. We consider the following standard search procedures from the neurally guided search literature:

- **random rollouts:** This search procedure uses the learned policy to sample a sequence of steps in the environment. Every rollout consists of N iterations; at each iteration an action is chosen according to π_θ . This action is executed in the environment by taking an environment step and the updated current geometry is presented back to the policy to sample the next action. N is capped to a fixed rollout length of $\max(\frac{f_p}{2}, 2)$, where f_p is the number of planar faces in G_t . If the agent

Table 3. Reconstruction results for IoU and exact reconstruction at 20 and 100 environment steps using random rollouts with different agents trained on human designed data. The best result in each column is shown in bold. Lower values are better for conciseness.

Agent	IoU		Exact Recon. %		Concise.
	20 Steps	100 Steps	20 Steps	100 Steps	
gat	0.8742	0.9128	0.6191	0.6742	1.0206
gcn	0.8644	0.9042	0.6232	0.6754	1.0168
gin	0.8346	0.8761	0.5901	0.6301	1.0042
mlp	0.8274	0.8596	0.5658	0.5965	0.9763
rand	0.6840	0.8386	0.4157	0.5380	1.2824

fails to recover the target geometry in the current roll-out, we restart with a new roll-out and repeat the process.

- **beam search:** We rollout in parallel the top-k (where k is the beam width) candidate construction sequences for N iterations. Each sequence is ranked by the generation probability under $\pi_\theta, P_\theta(a_1 \dots a_r)$:

$$P_\theta(a_1 \dots a_r) = \prod_{i=1 \dots r} \pi_\theta(a_i | G_i, G_t)$$

At each iteration, we consider all possible extensions to the top-k candidates by one action under π_θ , and re-rank the extended candidate sequences under P_θ , keeping the top-k extended candidates. Then, for each of the k extended sequences, we execute a step in the environment to obtain the updated current geometries. Each run of the beam search results in kN environment steps. If the current k sequences reaches the rollout length without recovering the target geometry, the beam search restarts with the beam width doubled, allowing it to search a wider range of candidates.

- **best first search:** This search procedure explores the search space by maintaining a priority queue of candidate sequences, where the priority is ordered by P_θ . At each iteration, we dequeue the top candidate sequence and extend it by one action under π_θ , and these extended sequences are added back to the queue. An environment step is taken in a lazy fashion when the top candidate sequence is dequeued, and not when the extended sequences are added back to the queue. This process continues until the dequeued top candidate recovers the target geometry.

6 EVALUATION

We proposed a general strategy consisting of neurally guided search, powered by a neural-network trained via imitation on human designed, synthetic, and augmented data. To justify this strategy, we perform ablation studies, comparing our approach against a set of baselines on the *Fusion 360 Gallery* reconstruction dataset. We seek to answer the following:

- How do different neural representations, when used to represent the agent’s policy π_θ , perform on the CAD reconstruction task?

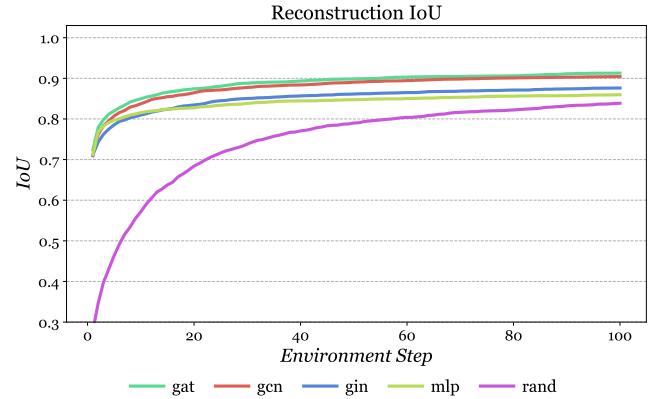


Fig. 11. Reconstruction IoU over 100 environment steps using random rollouts with different agents trained on human designed data.

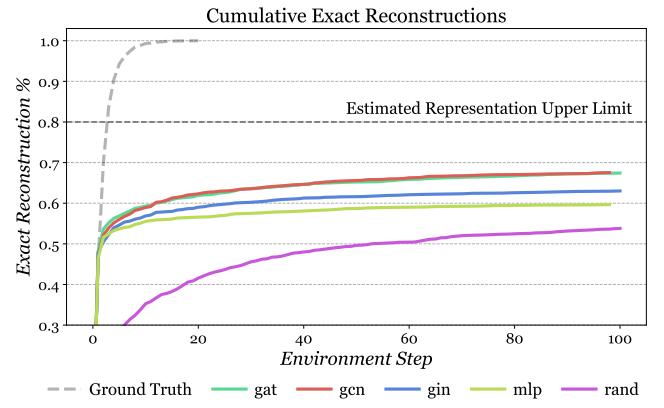


Fig. 12. Cumulative exact reconstructions over 100 environment steps using random rollouts with different agents trained on human designed data. The estimated upper limit of the face extrusion action representation is shown at 0.8.

- How does training a neural policy under human designed, synthetic, and augmented data affect CAD reconstruction performance?
- How do different neurally guided search procedures from the literature perform on the CAD reconstruction task?

For evaluation, we track the best IoU the agent has discovered so far, and whether exact reconstruction is achieved as a function of environment steps. We cap the total search budget to 100 steps to reflect a real world scenario. For experiments using human design data we train on the 59.2% of the training set that can be directly converted to a face extrusion sequence. We evaluate on the full test set in all cases. We estimate that approximately 80% of designs in our dataset can be reconstructed by finding alternative face extrusion sequences and note this when reporting exact reconstruction results.

6.1 Comparing Different Neural Representations

We evaluate five different kinds of neural network representations for π_θ to understand how different networks perform on the CAD reconstruction task. The **rand** agent uniformly samples from the available actions to serve as a naive baseline without any learning. **mlp** is a simple agent using a MLP that does not take advantage of message passing via graph topology. **gcn**, **gin**, and **gat** are MPN agents that use a Graph Convolution Network [Kipf and Welling 2016], Graph Isomorphism Network [Xu et al. 2018], and Graph Attention Network [Veličković et al. 2017] respectively. We use two MPN layers for all comparisons, with standard layer settings as described in Section A.3.2 of the appendix.

We report the reconstruction IoU and exact reconstructions using random rollout search for each agent as a function of the number of environment steps in Figure 11 and 12 respectively. We detail the exact results at step 20 and 100 in Table 3. Step 20 represents the point where it is possible to perform exact reconstructions for all designs in the test set. We also detail the conciseness of the recovered sequence for exact reconstructions. We note that all neurally guided agents outperform the random agent baseline. The topology information available with a MPN is found to improve reconstruction performance. The **gat** and **gcn** agents show the best performance but fall well short of exact reconstruction on all designs in the test set, demonstrating that the CAD reconstruction task is non-trivial and an open problem for future research.

6.2 Comparing Human and Synthetic Data Performance

We evaluate four **gcn** agents trained on different data sources to understand how synthetic data performs compared to human design data. **real** is trained on the standard human design training set. **syn** is trained on synthetic data from procedurally generated sketches of rectangles and circles extruded randomly (Figure 15, top). Leveraging basic primitives is a common method to generate synthetic data for program synthesis [Ellis et al. 2019; Li et al. 2020b; Sharma et al. 2017], that typically results in less sophisticated designs compared to human design data. **semi-syn** is trained on semi-synthetic designs that use existing sketches in the training set with two or more extrude operations to match the distribution of the number of faces in the training set (Figure 15, bottom). This approach results in more complex designs than the pure synthetic designs. We deliberately use these two approaches for data generation to better compare human design data to synthetic data in different distributions. **aug** is trained on the human design training set mixed with additional semi-synthetic data. We hold the training data quantity constant across agents, with the exception of the **aug** agent that contains a larger quantity from two sources. All agents are evaluated on the standard human design test set.

Figure 13 and 14 show that training on human design data offers a significant advantage over synthetic and semi-synthetic data for reconstruction IoU and exact reconstructions respectively. For the **aug** agent reconstruction performance is aided early on by data augmentation. We attribute this early performance improvement to semi-synthetic designs with 1 or 2 extrusions appearing similar to human designs. Conversely, we observe that semi-synthetic designs with multiple randomly applied extrusions appear less and

Table 4. Reconstruction results for IoU and exact reconstruction at 20 and 100 environment steps using random rollouts and gcn agents trained on human designed data (real), a mixture of human designed and semi-synthetic data (aug), semi-synthetic data (semi-syn), and synthetic data (syn). The best result in each column is shown in bold. Lower values are better for conciseness.

Agent	IoU		Exact Recon. %		Concise.
	20 Steps	100 Steps	20 Steps	100 Steps	
real	0.8644	0.9042	0.6232	0.6754	1.0168
aug	0.8707	0.8928	0.6452	0.6701	0.9706
semi-syn	0.8154	0.8473	0.5780	0.6104	1.0070
syn	0.6646	0.7211	0.4383	0.4835	1.0519

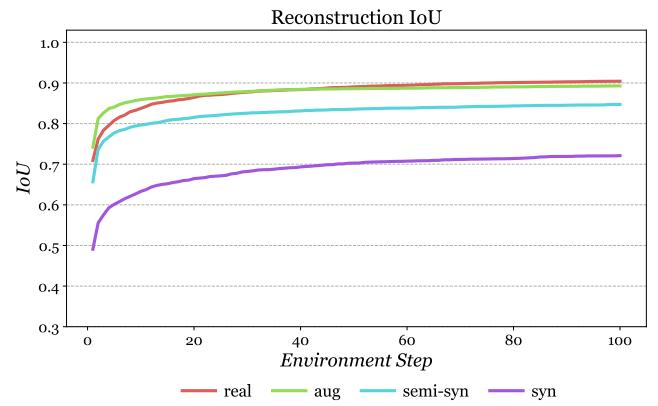


Fig. 13. Reconstruction IoU over 100 environment steps using random rollouts and gcn agents trained on human designed data (real), a mixture of human designed and semi-synthetic data (aug), semi-synthetic data (semi-syn), and synthetic data (syn).

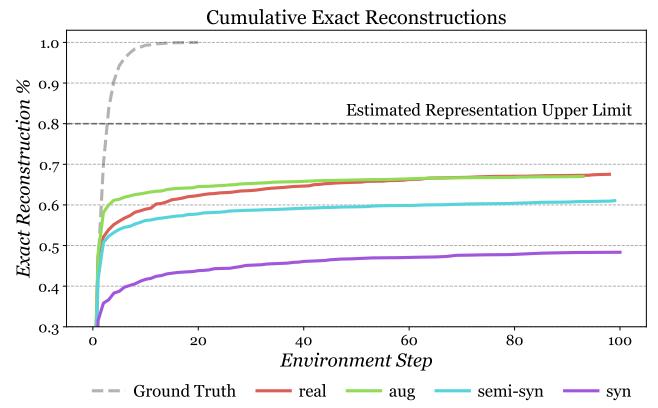


Fig. 14. Cumulative exact reconstructions over 100 environment steps using random rollouts and gcn agents trained on human designed data (real), a mixture of human designed and semi-synthetic data (aug), semi-synthetic data (semi-syn), and synthetic data (syn). The estimated upper limit of the face extrusion action representation is shown at 0.8.

less similar to human design due to the random composition of

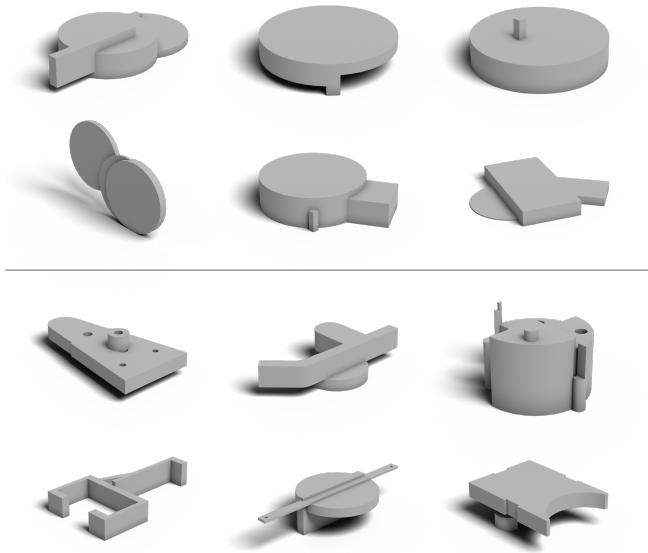


Fig. 15. Top: example *synthetic* data created by extruding circles and rectangles. Bottom: example *semi-synthetic* data created by extruding human designed sketches.

Table 5. Reconstruction results for IoU and exact reconstruction at 20 and 100 environment steps using gcn agents with best first search (best), random rollout search (rand) and beam search (beam). The best result in each column is shown in bold. Lower values are better for conciseness.

Agent	IoU		Exact Recon. %		Concise.
	20 Steps	100 Steps	20 Steps	100 Steps	
best	0.8831	0.9186	0.5971	0.6348	0.9215
rand	0.8644	0.9042	0.6232	0.6754	1.0168
beam	0.8640	0.8982	0.5739	0.6122	0.9275

extrusions. This difference in distribution between human and synthetic designs becomes more prevalent as search progresses. Table 4 provides exact results at environment step 20 and 100.

6.3 Qualitative Results

Figure 18 shows a visualization of ground truth construction sequences compared with the reconstruction results from other agents using random search. The rollout with the highest IoU is shown with the IoU score and total environment steps taken. Steps that don’t change the geometry or occur after the highest IoU are omitted from the visualization.

6.4 Comparing Search Procedures

We compare the effects of three different search procedures from the neurally guided search literature. Here, **rand** is random rollout, **beam** is beam search, and **best** is best-first search. For each search algorithm we use the **gcn** agent described in Section 6.1 trained on the standard human design training set. Figure 16, 17, and Table 5 show that all three search algorithms perform similarly for reconstruction IoU, while rand performs best for exact reconstruction.

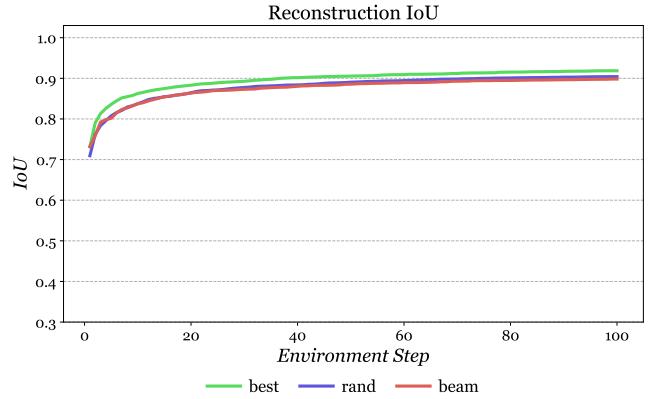


Fig. 16. Reconstruction IoU over 100 environment steps using the gcn agent with best first search (best), random rollout search (rand) and beam search (beam).

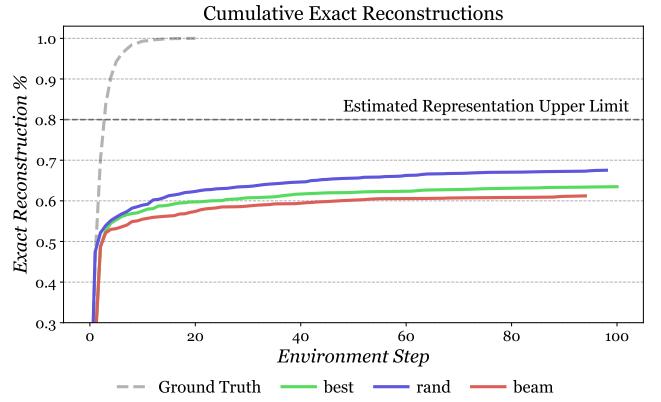


Fig. 17. Cumulative exact reconstructions over 100 environment steps using the gcn agent with best first search (best), random rollout search (rand) and beam search (beam). The estimated upper limit of the face extrusion action representation is shown at 0.8.

The performance of rand for exact reconstruction can be explained by the limited search budget of 100 environment steps: the rand algorithm is more likely to sample distinct sequences for a small number of samples, whereas beam will sample half its sequences identical to the previous rounds before the doubling, and best might not be sampled enough to explore a sequence long enough to contain the correct program.

We expect beam and best to outperform rand as the number of search budget increases, similar to Ellis et al. [2019]. However, the limitation of the search budget is important, as each design in our test set takes between 5-35 seconds to reconstruct on average. The majority of evaluation time is spent inside the *Fusion 360 Gym* executing modeling operations and graph generation, both computationally expensive yet crucial operations that must be taken during reconstruction.

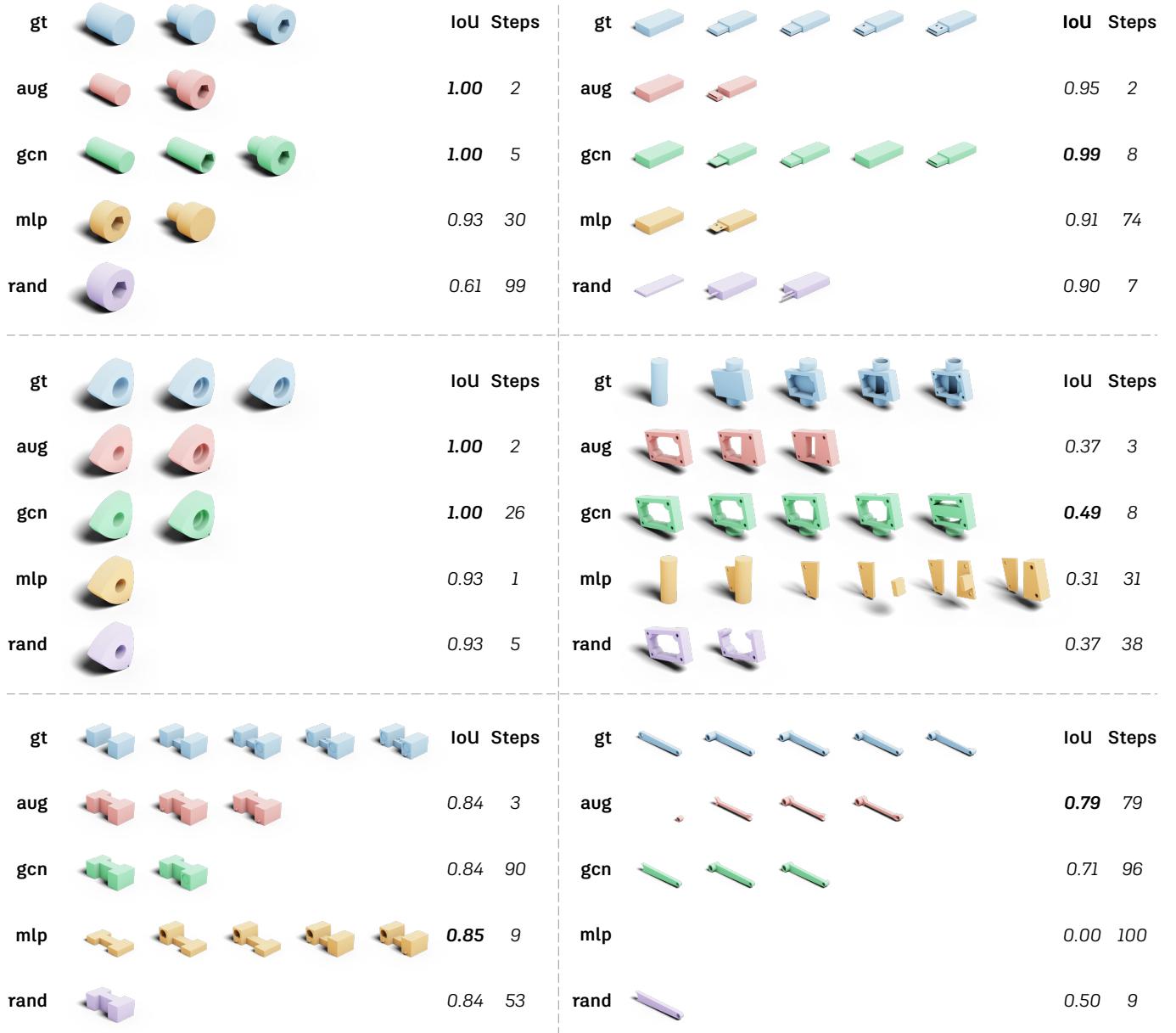


Fig. 18. Qualitative construction sequence results comparing the ground truth (gt) to reconstructions using different agents with random rollout search.

6.5 Discussion

For practical application of CAD reconstruction it is necessary to have an exact reconstruction where all details of a design are reconstructed in a concise way. It is notable that incorrect reconstructions can score well with the IoU metric, but omit important design details. For example, the small holes in the USB connector in Figure 18b are omitted from the gcn reconstruction. We suggest IoU should be a secondary metric, with future work focusing on improving exact reconstruction performance with concise construction sequences.

Conciseness should always be considered alongside exact reconstruction performance as naive approaches that only reconstruct short sequences can achieve good conciseness scores.

7 CONCLUSION AND FUTURE DIRECTIONS

In this paper we presented the *Fusion 360 Gallery* reconstruction dataset and environment for learning CAD reconstruction from sequential 3D CAD data. We outlined a standard CAD reconstruction task, together with evaluation metrics, and presented results from a neurally guided search approach.

7.1 Limitations

Our dataset contains only designs created using *sketch* and *extrude* rather than the full array of CAD modeling operations. Short construction sequences make up a sizable portion of the data: 3267/8625 (38%) of designs have only a single extrude operation. From the single extrude designs, some exhibit more complexity: 347 have >1 sketch profile resulting in ≥ 1 bodies from a single extrude operation, and 998 have ≥ 8 sketch curves. Other designs are washers, pegs, and plates, common in mechanical CAD assemblies. We avoid filtering simple designs to ensure the dataset is representative of user-designed CAD. Spline curves represent 4% of curves in the dataset and are not currently supported by our high-level DSL, however they can be reconstructed via the `reconstruct_curve()` command (Section A.2.1).

The success of the `rand` agent demonstrates that short construction sequences can be solved by a naive approach. This is due to several factors: 1) our action representation that uses B-Rep faces, 2) our search procedure discarding invalid actions, and 3) designs in the dataset with a low number of planar faces and extrude steps. For example, a washer has four B-Rep faces (*planar-top*, *cylinder-inside*, *cylinder-outside*, *planar-bottom*), giving the random agent a 2/2 chance of success as either *planar-top* \rightarrow *planar-bottom*, or vice versa, are considered correct and extrusions from non-planar faces are invalid. Although the random agent can achieve moderate success with simple designs, the problem quickly becomes challenging for more complex designs. All agents struggle to achieve exact reconstructions within the search budget for construction sequence lengths ≥ 4 .

7.2 Future Work

Future extensions of this work include *sample efficient* search strategies to ensure successful recovery of construction sequences with fewer interactions with the environment and leveraging constraints present in the dataset to guide CAD program synthesis. More broadly we envision the dataset can aid the creation of 3D geometry using the same CAD modeling operations as human designers, exploiting the knowledge of domain experts on how shapes are defined and leveraging the strengths of industrial CAD modeling software. By learning to translate point cloud, image, or mesh data into a sequence of high level modeling operations [Ellis et al. 2018; Tian et al. 2019], watertight CAD models may be synthesized, providing an alternative approach to the reverse engineering problem [Buonomamici et al. 2018]. A remaining challenge is to develop representations that can be conditioned on the design geometry and topology created so far, leveraging the sequential nature of the data for self-attention [Vaswani et al. 2017]. Finally, beyond the simplified design space of sketch and extrude lies the full breadth of rich sequential CAD modeling operations.

REFERENCES

- Silvia Ansaldi, Leila De Floriani, and Bianca Falcidieno. 1985. Geometric modeling of solid objects by using a face adjacency graph representation. *ACM SIGGRAPH Computer Graphics* 19, 3 (1985), 131–139.
- Autodesk. 2012. *Inventor Feature Recognition*. <https://apps.autodesk.com/INVNTOR/en/Detail/Index?id=9172877436288348979>
- Autodesk. 2014. *Fusion 360 API*. <http://help.autodesk.com/view/fusion360/ENU/?guid=GUID-7B5A90C8-E94C-48DA-B16B-430729B734DC>
- Autodesk. 2015. *Autodesk Online Gallery*. <https://gallery.autodesk.com>
- Pal Benko, Geza Kos, Pál Benkő, Laszlo Andor, Géza Kós, Tamas Varady, László Andor, and Ralph Martin. 2002. Constrained Fitting in Reverse Engineering.
- Suzanne Fox Buchele. 2000. Three-dimensional binary space partitioning tree and constructive solid geometry tree construction from algebraic boundary representations. (2000).
- Suzanne F Buchele and Richard H Crawford. 2003. Three-dimensional halfspace constructive solid geometry tree construction from implicit boundary representations. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*. 135–144.
- Suzanne F Buchele and Angela C Roles. 2001. Binary space partitioning tree and constructive solid geometry representations for objects bounded by curved surfaces.. In *CCCG*. Citeseer, 49–52.
- Francesco Buonomamici, Monica Carfagni, Rocco Furferi, Lapo Governi, Alessandro Lapini, and Yary Volpe. 2018. Reverse engineering modeling methods and tools: a survey. *Computer-Aided Design and Applications* 15, 3 (2018), 443–464. <https://doi.org/10.1080/16864360.2017.1397894>
- Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. 2015. ShapeNet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015).
- Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2020. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 45–54.
- Dassault. 2019. *Solidworks FeatureWorks*. https://help.solidworks.com/2019/english/SolidWorks/fworks/c_Overview_of_FeatureWorks.htm
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
- Jacob Devlin, Jonathan Usseto, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. 2017. Robustfill: Neural program learning under noisy i/o. *arXiv preprint arXiv:1703.07469* (2017).
- Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. 2018. Inversecsg: Automatic conversion of 3d models to csg trees. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–16.
- Mathias Eitz, James Hays, and Marc Alexa. 2012. How do humans sketch objects? *ACM Transactions on graphics (TOG)* 31, 4 (2012), 1–10.
- Kevin Ellis, Maxwell Nye, Yewen Pu, Felix Sosa, Josh Tenenbaum, and Armando Solar-Lezama. 2019. Write, execute, assess: Program synthesis with a repl. In *Advances in Neural Information Processing Systems*. 9169–9178.
- Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. 2018. Learning to Infer Graphics Programs from Hand-Drawn Images. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc., 6059–6068. <https://proceedings.neurips.cc/paper/2018/file/678076842014c83cedab6b0ba0314-Paper.pdf>
- Pierre-Alain Fayolle and Alexander Pasko. 2016. An evolutionary approach to the extraction of object construction trees from 3D point clouds. *Computer-Aided Design* 74 (2016), 1–17.
- Markus Friedrich, Pierre-Alain Fayolle, Thomas Gabor, and Claudia Linnhoff-Popien. 2019. Optimizing evolutionary CSG tree extraction. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1183–1191.
- Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. 2019. SDM-NET: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–15.
- Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*. PMLR, 1263–1272.
- Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, Sylvia Pont, Fredo Durand, and Adrien Bousseau. 2019. Opensketch: A richly-annotated dataset of product design sketches. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 232.
- Karim Hamza and Kazuhiro Saitou. 2004. Optimization of constructive solid geometry via a tree-based multi-objective genetic algorithm. In *Genetic and Evolutionary Computation Conference*. Springer, 981–992.
- Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph Lambourne, Thomas Davies, Hooman Shayani, and Nigel Morris. 2020. UV-Net: Learning from Curve-Networks and Solids. *arXiv preprint arXiv:2006.10211* (2020).
- R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy Mitra, and Daniel Ritchie. 2020. ShapeAssembly: Learning to Generate Programs for 3D Shape Structure Synthesis. *ACM Transactions on Graphics (TOG)*, *Siggraph Asia 2020* 39, 6 (2020), Article 234.
- Ashwin Kalyan, Abhishek Mohta, Oleksandr Polozov, Dhruv Batra, Prateek Jain, and Sumit Gulwani. 2018. Neural-guided deductive search for real-time program synthesis from examples. *arXiv preprint arXiv:1804.01186* (2018).
- Kacper Kania, Maciej Zięba, and Tomasz Kajdanowicz. 2020. UCSG-Net-Unsupervised Discovering of Constructive Solid Geometry Tree. *arXiv preprint arXiv:2006.09102*

- (2020).
- Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. 2020. A Large-scale Annotated Mechanical Components Benchmark for Classification and Retrieval Tasks with Deep Neural Networks. In *Proceedings of 16th European Conference on Computer Vision (ECCV)*.
- Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. 2018. Neural relational inference for interacting systems. In *International Conference on Machine Learning*. PMLR, 2688–2697.
- Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A big CAD model dataset for geometric deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9601–9611.
- Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. 2020b. Sketch2CAD: Sequential CAD Modeling by Sketching in Context. *ACM Trans. Graph. (Proceedings of SIGGRAPH Asia 2020)* 39, 6 (2020), 164:1–164:14. <https://doi.org/10.1145/3414685.3417807>
- Jun Li, Chengjie Niu, and Kai Xu. 2020a. Learning part generation and assembly for structure-aware shape synthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 11362–11369.
- Renjie Liao, Yujia Li, Yang Song, Sherlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. 2019. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems*. 4255–4265.
- Cheng Lin, Tingxiang Fan, Wenping Wang, and Matthias Nießner. 2020. Modeling 3D Shapes by Reinforcement Learning. *ECCV* (2020).
- Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J. Mitra, and Leonidas J. Guibas. 2019a. StructureNet: Hierarchical Graph Networks for 3D Shape Generation. *ACM Trans. Graph.* 38, 6, Article 242 (Nov. 2019), 19 pages. <https://doi.org/10.1145/3355089.3356527>
- Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. 2019b. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 909–918.
- Chandrakana Nandi, Anat Caspi, Dan Grossman, and Zachary Tatlock. 2017. Programming language tools and techniques for 3D printing. In *2nd Summit on Advances in Programming Languages (SNAPL 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
- Chandrakana Nandi, James R Wilcox, Pavel Panckehka, Taylor Blau, Dan Grossman, and Zachary Tatlock. 2018. Functional programming for compiling and decompiling computer-aided design. *Proceedings of the ACM on Programming Languages 2, ICFP (2018)*, 1–31.
- Chandrakana Nandi, Max Willsey, Adam Anderson, James R. Wilcox, Eva Darulova, Dan Grossman, and Zachary Tatlock. 2020. Synthesizing Structured CAD Models with Equality Saturation and Inverse Transformations. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*. 31–44.
- Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. 2020. Polygen: An autoregressive generative model of 3d meshes. In *International Conference on Machine Learning*. PMLR, 7220–7229.
- Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. 2016. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.
- Nadav Schor, Oren Katzir, Hao Zhang, and Daniel Cohen-Or. 2019. CompoNet: Learning to generate the unseen by part synthesis and composition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 8759–8768.
- Adriana Schulz, Ariel Shamir, David I. W. Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. 2014. Design and Fabrication by Example. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2014)* 33, 4 (2014).
- Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. 2020. SketchGraphs: A Large-Scale Dataset for Modeling Relational Geometry in Computer-Aided Design. In *ICML 2020 Workshop on Object-Oriented Learning*.
- Jami J Shah, David Anderson, Yong Se Kim, and Sanjay Joshi. 2001. A discourse on geometric feature recognition from CAD models. *J. Comput. Inf. Sci. Eng.* 1, 1 (2001), 41–51.
- Vadim Shapiro and Donald L Vossler. 1993. Separation for boundary to CSG conversion. *ACM Transactions on Graphics (TOG)* 12, 1 (1993), 35–55.
- Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. 2017. CSGNet: Neural Shape Parser for Constructive Solid Geometry. CoRR abs/1712.08290 (2017). *arXiv preprint arXiv:1712.08290* (2017).
- Binil Starly. 2020. *FabWave - 3D Part Repository*. <https://www.dimelab.org/fabwave>
- O. Stava, S. Pirk, J. Kratt, B. Chen, R. Mundefinedch, O. Deussen, and B. Benes. 2014. Inverse Procedural Modelling of Trees. *Comput. Graph. Forum* 33, 6 (Sept. 2014), 118–131. <https://doi.org/10.1111/cgf.12282>
- Minhyuk Sung, Hao Su, Vladimir G Kim, Siddhartha Chaudhuri, and Leonidas Guibas. 2017. ComplementMe: Weakly-supervised component suggestions for 3D modeling. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–12.
- Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. 2011. Metropolis Procedural Modeling. *ACM Trans. Graph.* 30, 2, Article 11 (April 2011), 14 pages. <https://doi.org/10.1145/1944846.1944851>
- Yunhai Tang, Shipra Agrawal, and Yuri Faenza. 2019. Reinforcement learning for integer programming: Learning to cut. *arXiv preprint arXiv:1906.04859* (2019).
- Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. 2019. Learning to Infer and Execute 3D Shape Programs. In *International Conference on Learning Representations*.
- Carlos A. Vanegas, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Paul Waddell. 2012. Inverse Design of Urban Procedural Models. *ACM Trans. Graph.* 31, 6, Article 168 (Nov. 2012), 11 pages. <https://doi.org/10.1145/2366145.2366187>
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., 5988–6008. <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fb053c1c4a845aa-Paper.pdf>
- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- K.J. Weiler. 1986. *Topological structures for geometric modeling*. University Microfilms.
- Daniel Weiss. 2009. *Geometry-based structural optimization on CAD specification trees*. Ph.D. Dissertation. ETH Zurich.
- Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. 2020. Pq-net: A generative part seq2seq network for 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 829–838.
- Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.
- Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
- Zhibo Zhang, Prakhar Jaiswal, and Rahul Rai. 2018. FeatureNet: machining feature recognition based on 3D convolution neural network. *Computer-Aided Design* 101 (2018), 12–22.
- Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).
- Chuhang Zou, Ersin Yumer, Jimie Yang, Duygu Ceylan, and Derek Hoiem. 2017. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 900–909.

A APPENDIX

A.1 Fusion 360 Gallery Reconstruction Dataset

In this section we provide additional details on the *Fusion 360 Gallery* reconstruction dataset.

A.1.1 Data Processing. To process the data we use the Fusion 360 Python API to parse the native Fusion 360 .f3d files. Figure 19 shows an example assembly that is split up to produce multiple designs with independent construction sequences. The rounded edges are removed by suppressing fillets in the parametric CAD file. During processing color and material information is also removed.

After each construction sequence has been extracted we perform reconstruction and compare the reconstructed design to the original to ensure data validity. Failure cases and any duplicate designs, are not included in the dataset. We consider a design a duplicate when there is an exact match in all of the following: body count, face count, surface area to one decimal point, volume to one decimal point, and for each extrude in the construction sequence: extrude profile count, extrude body count, extrude face count, extrude side face count, extrude end face count, and extrude start face count. This process allows us to match designs that have been translated or rotated, while considering designs unique if they have matching geometry but different construction sequences. Duplicates account for approximately 5,000 designs. Figure 20 shows a random sampling of designs from the reconstruction dataset.

A.1.2 Geometry Data Format. As described in Section 3.1, we provide geometry in several data formats that we provided additional details on in this section.

Boundary Representation. A B-Rep consists of faces, edges, loops, coedges and vertices [Weiler 1986]. A face is a connected region of the model’s surface. An edge defines the curve where two faces meet and a vertex defines the point where edges meet. Faces have an underlying parametric surface which is divided into visible and hidden regions by a series of boundary loops. A set of connected faces forms a body.

B-Rep data is provided as .smt files representing the ground truth geometry and .step as an alternate neutral B-Rep file format. The .smt file format is the native format used by Autodesk Shape Manager, the CAD kernel within Fusion 360, and has the advantage of minimizing conversion errors.

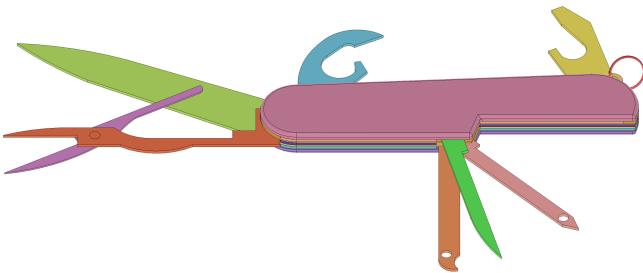


Fig. 19. An example multi-component assembly that is broken up into separate designs (highlighted with color), each with an independent construction sequence.

Mesh. Mesh data is provided in .obj format representing a triangulated version of the B-Rep. Each B-Rep face is triangulated separately and is therefore not manifold.

Other representations, such as point clouds or voxels, can be generated using existing data conversion routines and are not included in the dataset. For convenience we include a thumbnail .png image file together with each geometry.

Files are provided in a single directory, with a naming convention as follows: XXXXX_YYYYYYYY_ZZZZ[_1234].ext. Here XXXXX represents the project, YYYYYYYY the file, ZZZZ the component, and _1234 the extrude index. If _1234 is absent the file represents the final design.

A.1.3 Design Complexity. A key goal of the reconstruction dataset is to provide a suitably scoped baseline for learning-based approaches to CAD reconstruction. Restricting the modeling operations to *sketch* and *extrude* vastly narrows the design space and enables simpler shape grammars for reconstruction. Each design represents a component in Fusion 360 that can have multiple geometric bodies. Figure 21 (left) illustrates that the vast majority of designs have a single body. The number of B-Rep faces in each design gives a good indication of the complexity of the dataset. Figure 21 (right) shows the number of faces per design as a distribution, with the peak being between 5-10 faces per design. As we do not filter any of the designs based on complexity, this distribution reflects real designs where simple washers and flat plates are common components in mechanical assemblies.

A.1.4 Construction Sequence. The construction sequence is the series of *sketch* and *extrude* operations that are executed to produce the final geometry. We provide the construction sequence in a JSON format text file. Each step in the construction sequence has associated parameters that are stored in that entity. For example, *sketch* entities will store the curves that make up the sketch. Each construction sequence must have at least one *sketch* and one *extrude* step, for a minimum of two steps. The average number of steps is 4.74, the median 4, the mode 2, and the maximum 61. Figure 22 illustrates the distribution of construction sequence length and the most frequent construction sequence combinations.

With access to the full parametric history, it is possible to extract numerous relationships from the dataset that can be used for learning. Starting at a high level, we know the order of modeling operations in the construction sequence. The sketch geometry, B-Rep faces, and triangles derived from them, can be traced back to a position in the construction sequence. The type of geometry created by each modeling operation is also known. For example, sketches create trimmed profiles where the curves intersect to form closed loops; extrude operations produce B-Rep faces with information such as which faces were on the side or ends of an extrusion. In addition, the sequence of B-Rep models themselves contain valuable topology information that can be leveraged, such as the connectivity of B-Rep faces and edges. Finally geometric information like points and normal vectors can be sampled from the parametric surfaces. Feature diversity enables many different learning representations and architectures to be leveraged and compared.

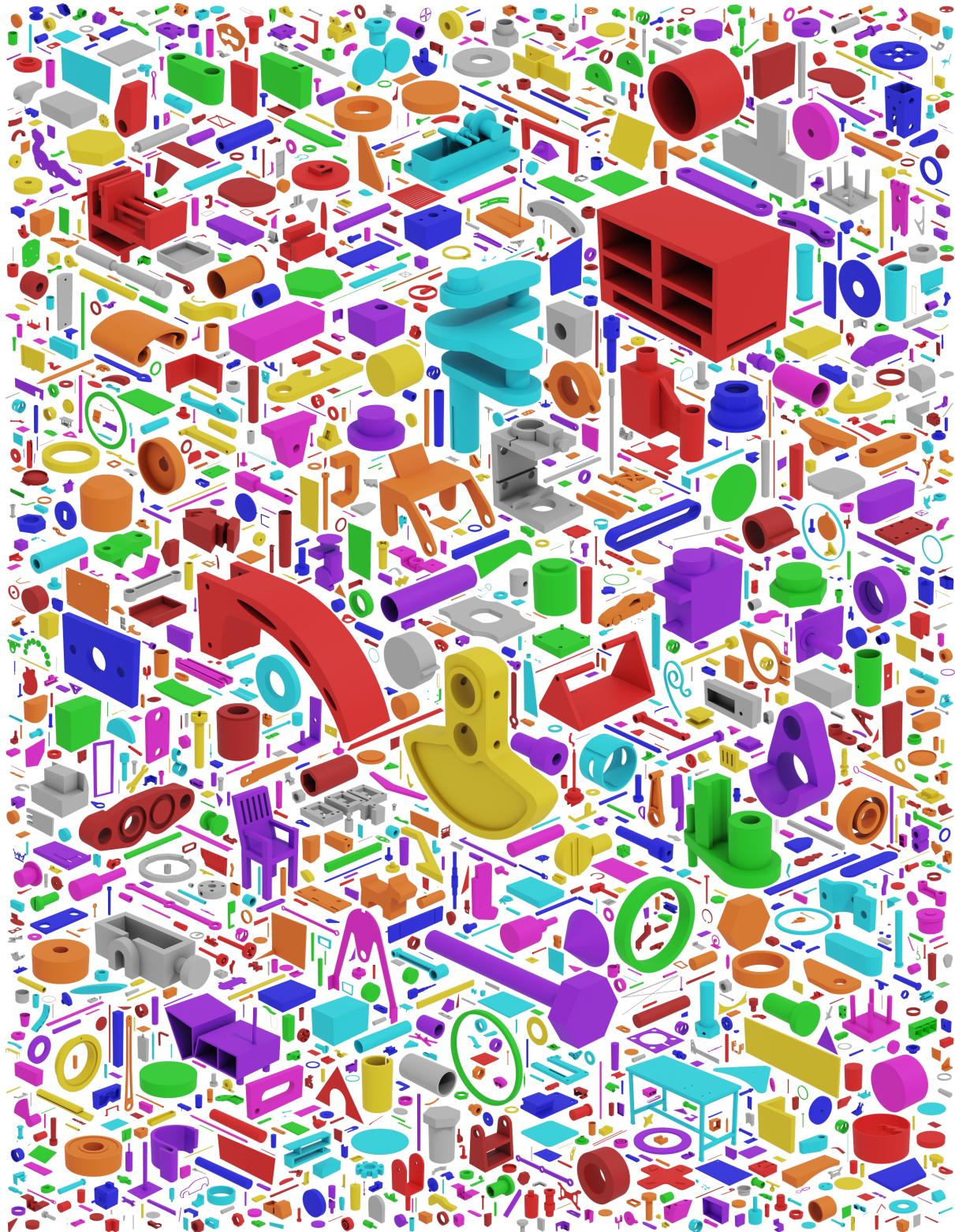


Fig. 20. A random sampling of designs from the *Fusion 360 Gallery* reconstruction dataset.

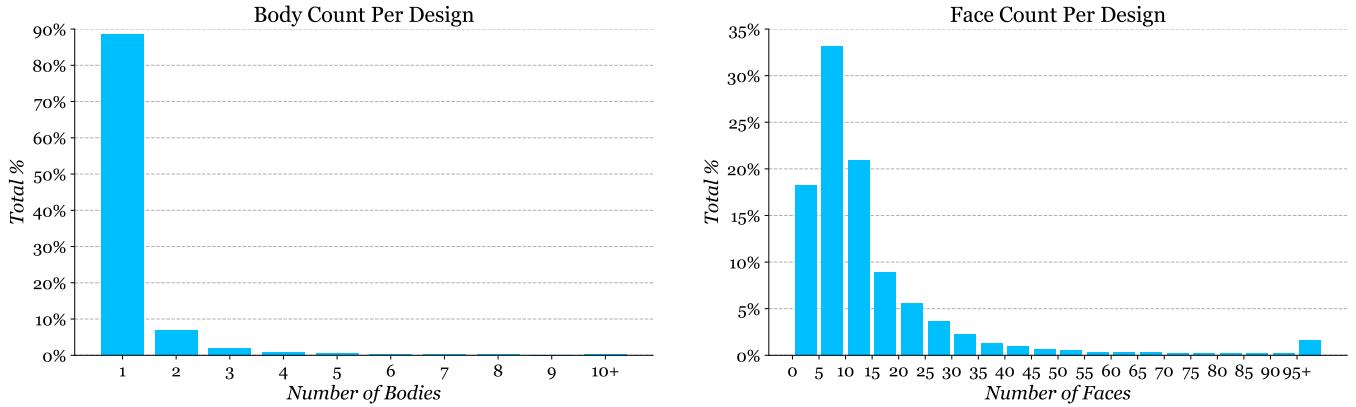


Fig. 21. Left: The number of bodies per design shown as a distribution. Right: The number of B-Rep faces per design shown as a distribution.

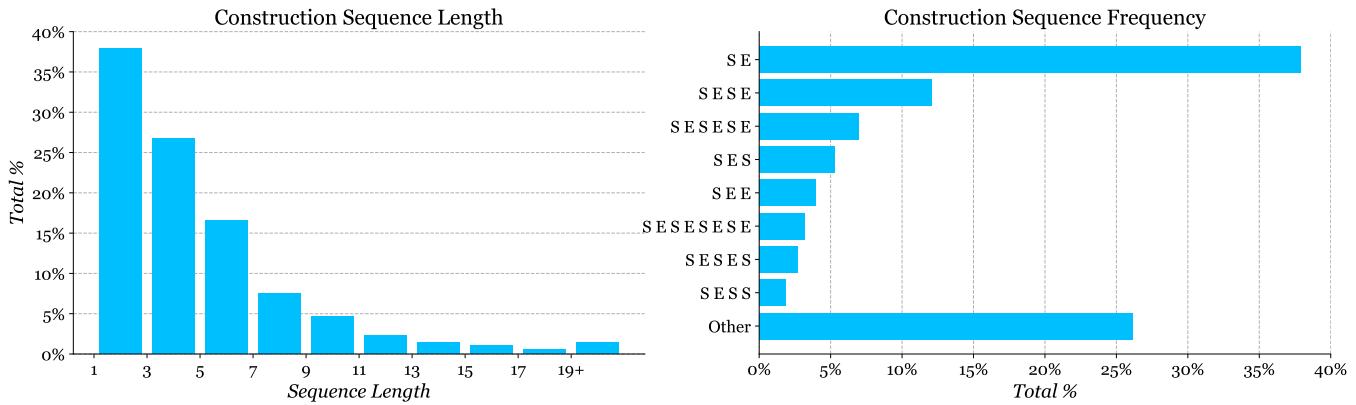


Fig. 22. Left: The distribution of construction sequence length. Right: The distribution of common construction sequences. S indicates a *Sketch* and E indicates an *Extrude* operation.

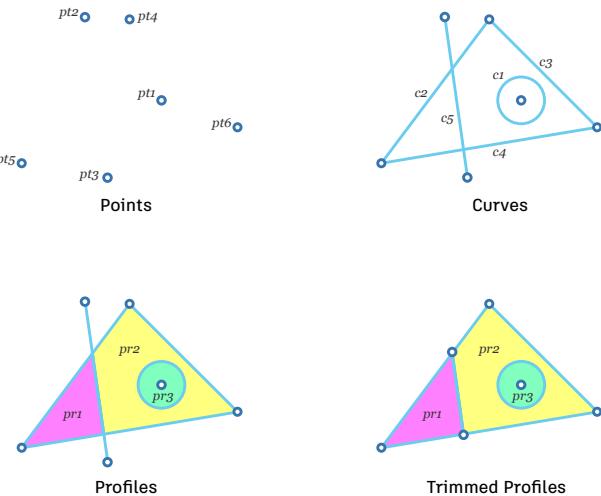


Fig. 23. Sketch primitives.

A.1.5 Sketch. In this section we describe the sketch data in further detail and present statistics illustrating the data distribution. Figure 23 illustrates the geometric 2D primitives, described in section 3.2, that make up a sketch. Sketches are represented as a series of points (*pt1*...*pt6*), that create curves (*c1*...*c5*), that in turn create profiles (*pr1*...*pr3*), illustrated with separate colors. Profiles also have a trimmed representation that contains only closed loops without open curves. The trimmed representation is shown in the lower right of Figure 23 where the *c5* is trimmed and incorporated into *pr1* and *pr2*.

Points. Each point is provided with a universally unique identifier (UUID) key and a Point3D data structure with *x*, *y*, and *z*. Sketch primitives are drawn in a local 2D coordinate system and later transformed into world coordinates. As such all sketch points have a *z* value of 0.

Curves. Each curve has a UUID key and a SketchCurve that can represent the curve types listed below. The parameters for each curve type can be referenced via the Fusion 360 API documentation linked below.

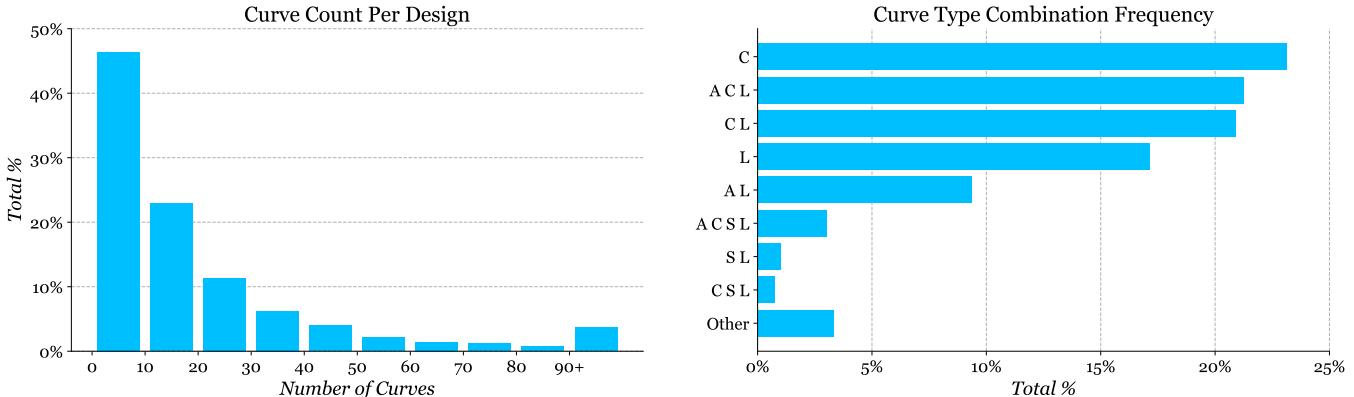


Fig. 24. Left: The number of curves in each design, shown as a distribution. Right: Common curve combinations in each design, shown as a distribution. Each curve type is abbreviated as follows: C - SketchCircle, A - SketchArc, L - SketchLine, S - SketchFittedSpline.

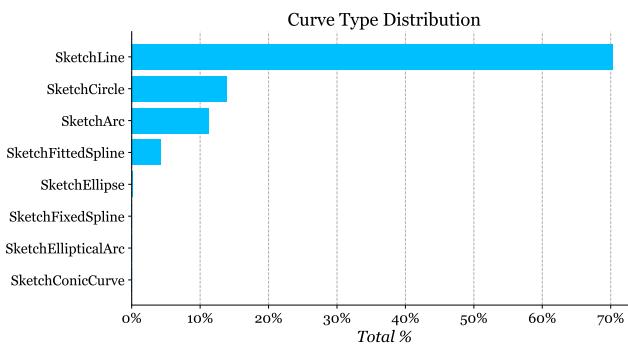


Fig. 25. The distribution of curve types.

- SketchArc
- SketchCircle
- SketchConicCurve
- SketchEllipse
- SketchEllipticalArc
- SketchFittedSpline
- SketchFixedSpline
- SketchLine

Figure 24 illustrates the distribution of curve count per design and the frequency that different curve combinations are used together in a design. Figure 25 shows the overall distribution of curve types in the dataset. It is notable that mechanical CAD sketches rely heavily on lines, circles, and arcs rather than spline curves.

Profiles. Profiles represent a collection of curves that join together to make a closed loop. In *Fusion 360* profiles are automatically generated from arbitrary curves that don't necessarily connect at the end points. In Figure 23 two profiles (*pr1* and *pr2*) are generated when the line crosses the triangle. We provide both the original curves (Figure 23, top right) used to generate the profiles (Figure 23, bottom left) and the trimmed profile information containing just the closed

profile loop (Figure 23, bottom right). Loops within profiles have a flag that can be set to specify holes.

Dimensions. User specified sketch dimensions are used to define set angles, diameters, distances etc. between sketch geometry to constraint the sketch as it is edited. Each dimension has a UUID key and a SketchDimension that can represent the dimension types listed below. Each dimension references one or more curves by UUID. The parameters for each dimension type can be referenced via the *Fusion 360 API* documentation linked below.

- SketchAngularDimension
- SketchConcentricCircleDimension
- SketchDiameterDimension
- SketchEllipseMajorRadiusDimension
- SketchEllipseMinorRadiusDimension
- SketchLinearDimension
- SketchOffsetCurvesDimension
- SketchOffsetDimension
- SketchRadialDimension

Constraints. Constraints define geometric relationships between sketch geometry. For example, a symmetry constraint enables the user to have geometry mirrored, or a parallel constraint ensures two lines are always parallel. Each constraint has a UUID key and a GeometricConstraint that can represent the constraint types listed below. Each constraint references one or more curves by UUID. The parameters for each constraint type can be referenced via the *Fusion 360 API* documentation linked below.

- CircularPatternConstraint
- CoincidentConstraint
- CollinearConstraint
- ConcentricConstraint
- EqualConstraint
- HorizontalConstraint
- HorizontalPointsConstraint

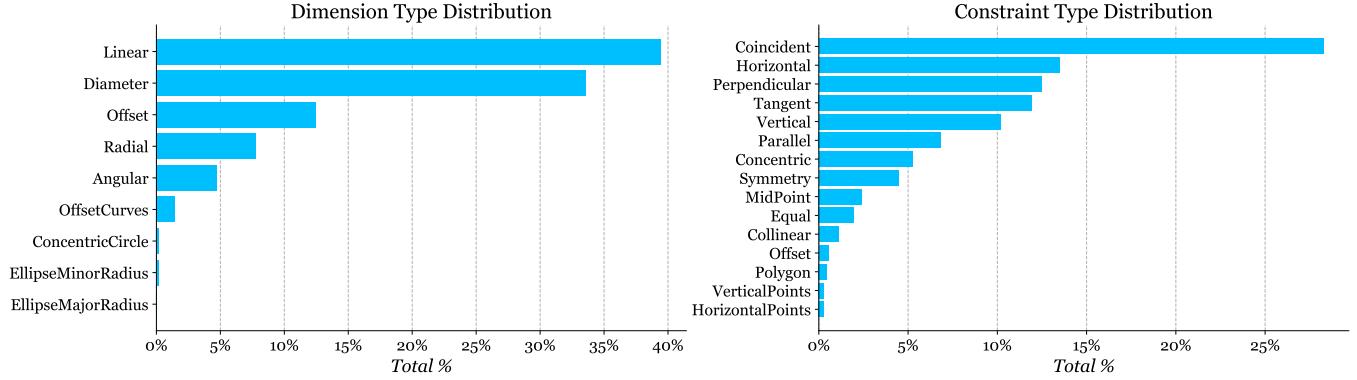


Fig. 26. The distribution of constraint (left) and dimension (right) types.

- MidPointConstraint
- OffsetConstraint
- ParallelConstraint
- PerpendicularConstraint
- PolygonConstraint
- RectangularPatternConstraint
- SmoothConstraint
- SymmetryConstraint
- TangentConstraint
- VerticalConstraint
- VerticalPointsConstraint

Figure 26 illustrates the distribution of dimension and constraint types in the dataset.

A.1.6 Extrude. In this section we describe the extrude data in further detail and present statistics illustrating the data distribution. Extrude operations have a number of parameters that are set by the user while designing. Figure 27 shows how a sketch (left) can be extruded a set distance on one side, symmetrically on two sides, with different distances on each side, as well as tapered. The first extrude operation of a construction sequence always creates a new body, with subsequent extrudes interacting with that body via Boolean operations.

Figure 28 outlines the distribution of different extrude types and operations. Note that tapers can be applied in addition to any extrude type, so the overall frequency of each is shown rather than a relative percentage.

A.2 Fusion 360 Gym

In this section we provide additional information about the functionality available in the *Fusion 360 Gym*. The *Fusion 360 Gym* requires the Autodesk Fusion 360 desktop CAD application, available on both macOS and Windows for free to the academic community. Although Fusion 360 is a cloud connected desktop application, the *Fusion 360 Gym* does all processing locally. The *Fusion 360 Gym* consists of a *server* that runs inside of Fusion 360 and receives commands from a *client* running externally. Multiple instances of the *Fusion 360*

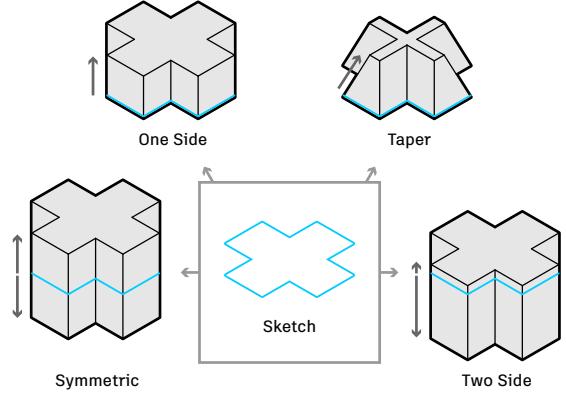


Fig. 27. An extrude can be expressed in several different ways: perpendicular from a sketch for a set distance along one side, a symmetrical distance along both sides, or separate distances along two sides. Additionally the extrude can be tapered at an angle.

Gym server can be run in parallel. The remainder of this section introduces the available commands from the *client*.

A.2.1 Reconstruction Commands. Reconstruction commands can reconstruct the existing designs at different granularity levels from json files provided with the *Fusion 360 Gallery* reconstruction dataset.

- `reconstruct(file)`: reconstruct an entire design from the provided json file.
- `reconstruct_sketch(sketch_data, sketch_plane, scale, translate, rotate)`: reconstruct a sketch from the provided sketch data. A `sketch_plane` can be either: (1) a string value representing a construction plane: XY, XZ, or YZ; (2) a B-Rep planar face id; or (3) a point3d on a planar face of a B-Rep.
- `reconstruct_profile(sketch_data, sketch_name, profile_id, scale, translate, rotate)`: reconstruct a single profile from the provided sketch data, a sketch name, and a profile id.
- `reconstruct_curve(sketch_data, sketch_name, curve_id, scale, translate, rotate)`: reconstruct a single curve from the provided sketch data, a sketch name, and a curve id.

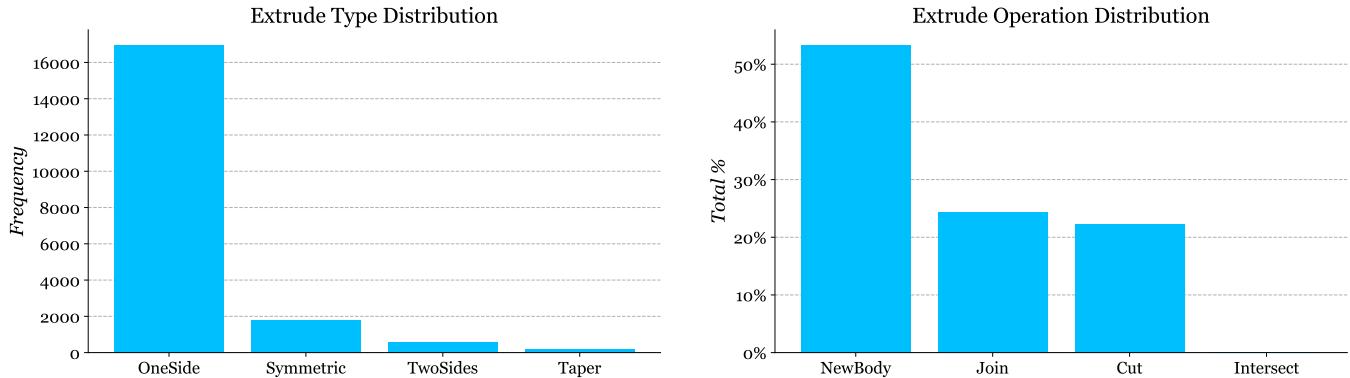


Fig. 28. The distribution of extrude types (left) and operations (right).

- `reconstruct_curves(sketch_data, sketch_name, scale, translate, rotate)`: reconstruct all curves from the provide sketch data and a sketch name.

A.2.2 Target Reconstruction Commands. Target reconstruction commands set the target design to be used with reconstruction.

- `set_target(file)`: set the target to be reconstructed with a .step or .smt file. The call returns a face adjacency graph representing the B-Rep geometry/topology and a `bounding_box` of the target that can be used for normalization.
- `revert_to_target()`: revert to the target design, removing all reconstruction geometry.

A.2.3 Sketch Extrusion Commands. Sketch extrusion commands allows users to incrementally create new designs by generating the underlying sketch primitives and extruding them by an arbitrary amount.

- `add_sketch(sketch_plane)`: add a sketch to the design. A `sketch_plane` can be either: (1) a string value representing a construction plane: XY, XZ, or YZ; (2) a B-Rep planar face id; or (3) a point3d on a planar face of a B-Rep.
- `add_point(sketch_name, p, transform)`: add a point to create a new sequential line in the given sketch. `p` is either a point in the 2D sketch space or a point in the 3D world coordinate space if `transform="world"` is specified.
- `add_line(sketch_name, p1, p2, transform)`: add a line to the given sketch. `p1` and `p2` are the same as defined in `add_point()`.
- `add_arc(sketch_name, p1, p2, angle, transform)`: add an arc to the given sketch. `p1` is the start point of the arc and `p2` is the center point of the arc. Other properties of `p1` and `p2` are the same as defined in `add_point()`. `angle` is the arc's angle, measured in degrees.
- `add_circle(sketch_name, p, radius, transform)`: add a circle to the given sketch. `p` is the center point of the circle. Other properties of `p` are the same as defined in `add_point()`. `radius` is the radius of the circle.
- `close_profile(sketch_name)`: close the current set of lines to create one or more profiles by joining the first point to the last point.
- `add_extrude(sketch_name, profile_id, distance, operation)`: add an extrude to the design. Four operations are supported:
 - `JoinFeatureOperation`

- `CutFeatureOperation`
- `IntersectFeatureOperation`
- `NewBodyFeatureOperation`

It returns a data structure with:

- `extrude`: B-Rep face information, including vertices, generated from the extrusion.
- `graph`: face adjacency graph of the current design in "PerFace" format.
- `bounding_box`: bounding box of the current design that can be used for normalization.
- `iou`: intersection over union result if a target design has been set with `set_target()`.

A.2.4 Face Extrusion Commands. Face extrusion commands enable a target design to be reconstructed using extrude operations from face to face.

- `add_extrude_by_target_face(start_face, end_face, operation)`: add an extrude between two faces of the target. Four operations are supported:
 - `JoinFeatureOperation`
 - `CutFeatureOperation`
 - `IntersectFeatureOperation`
 - `NewBodyFeatureOperation`
- `add_extrudes_by_target_face(actions, revert)`: execute multiple extrude operations, between two faces of the target, in sequence.

A.2.5 Randomized Reconstruction Commands. Randomized reconstruction commands allow users to sample designs, sketches, and profiles from existing designs in the *Fusion 360 Gallery* and support distribution matching of parameters, in support of generations of semi-synthetic data. Figure 29 shows example designs created using randomized reconstruction commands.

- `get_distributions_from_dataset(data_dir, filter, split_file)`: get a list of distributions from the provided dataset. The command currently supports the following distributions:
 - the starting sketch place

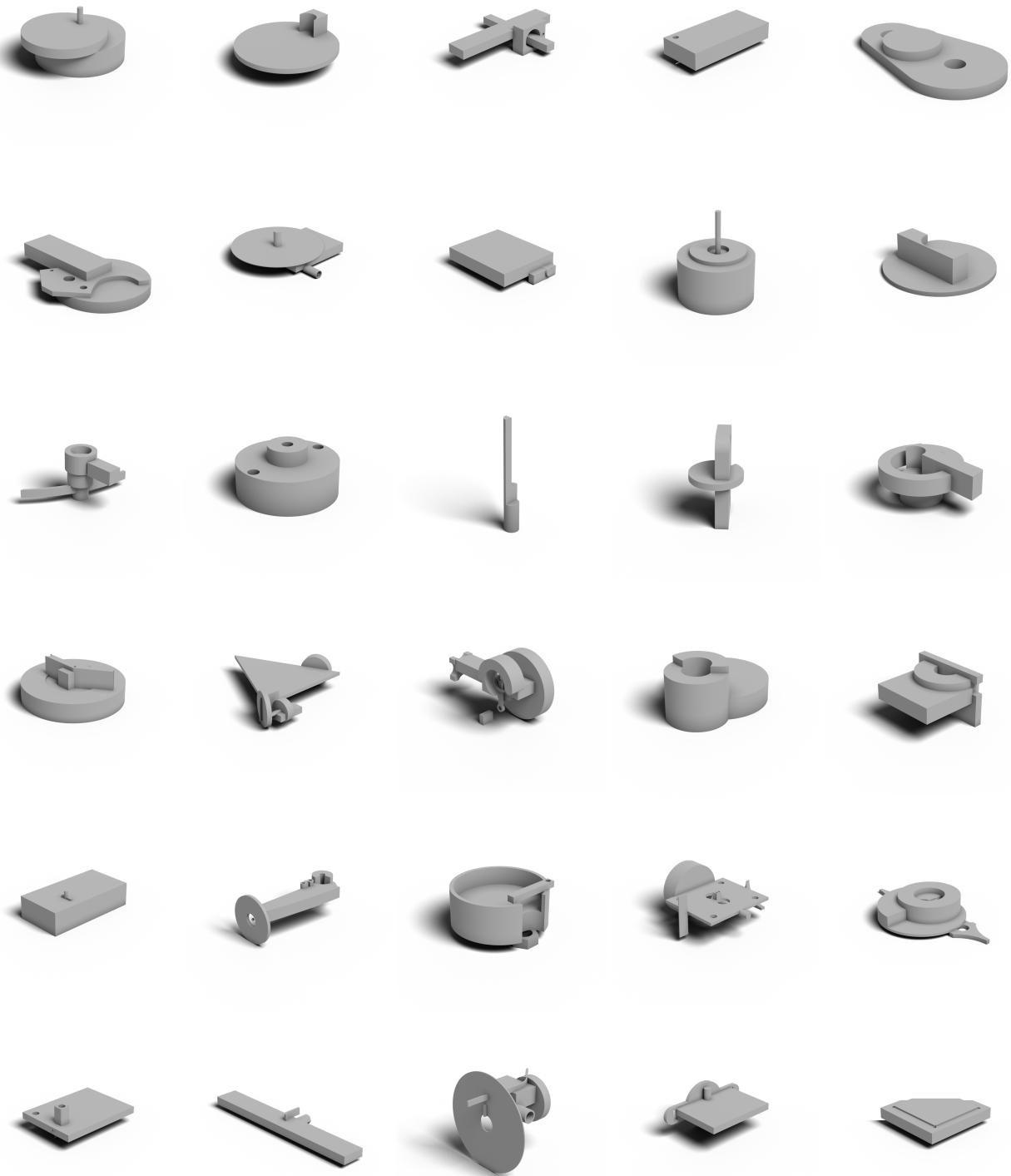


Fig. 29. Example designs created using randomized reconstruction commands.

- the number of faces
- the number of extrusions
- the length of sequences
- the number of curves
- the number of bodies
- the sketch areas
- the profile areas.
- `get_distribution_from_json(json_file)`: return a list of distributions saved in the given json file.
- `distribution_sampling(distributions, parameters)`: sample distribution matching parameters for one design from the distributions.
- `sample_design(data_dir, filter, split_file)`: randomly sample a json file from the given dataset.
- `sample_sketch(json_file, sampling_type, area_distribution)`: sample one sketch from the provided design. Three sampling types are provided:
 - `random`, return a sketch randomly sampled from the provided design.
 - `deterministic`, return the largest sketch in the design.
 - `distributive`, return a sketch that its area is in the distribution of the provided dataset.
- `sample_profiles(sketch_name, max_number_profiles, sampling_type, area_distribution)`: sample profiles from the provided sketch. Three sampling types are provided:
 - `random`, return profiles randomly sampled from the provided sketch.
 - `deterministic`, return profiles that are larger than the average area of the profiles in the sketch.
 - `distributive`, return profiles that the areas are in the distribution of the provided dataset.

A.2.6 Export Commands. Export commands enable the existing designs to be exported in the following formats:

- `mesh(file)`: retrieve a mesh in .obj or .stl format and write it to the local file provided.
- `brep(file)`: retrieve a brep in .step, .smi, or .f3d format and write it to a local file provided.
- `sketches(dir, format)`: retrieve each sketch in .png or .dxf format and write them to a local directory provided.
- `screenshot(file, width, height)`: retrieve a screenshot of the current design as a png image and write it to a local file provided.
- `graph(file, dir, format)`: retrieve a face adjacency graph in a given format and write it in a local directory provided.

A.3 CAD Reconstruction

In this section we provide additional details of the experiments performed on the CAD reconstruction task described in Section 5.

A.3.1 Data Preparation. The agents are trained on a subset of the reconstruction dataset that has been converted into a face extrusion sequence. Due to the simplified face extrusion representation, not all designs from the dataset can be converted to a face extrusion sequence. Figure 30 shows several common conversion limitations

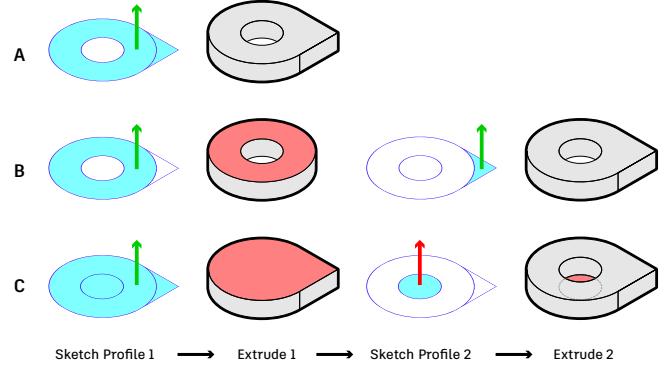


Fig. 30. Different construction sequences (A-C) for the same geometry. During conversion to a face extrusion sequence, the necessary face information (highlighted in red) does not exist in the target, meaning B and C can not be converted. Green arrows indicate *new body/join* extrude operations, while red arrows indicate *cut* extrude operations.

where necessary face information (highlighted in red) is not present in the target geometry. The intermediate top face in Figure 30 B disappears when merged with the top face of Extrude 2. In Figure 30 C a hole cut through the geometry means the intermediate top face of Extrude 1 is absent and there is no start or end face in the target geometry to perform the cut operation used in Extrude 2. Although it is possible to find alternate face extrusion sequences with heuristic rules, we instead try to maintain the user designed sequence with the exception of reversing the direction of the extrusion in some scenarios, e.g. the end face becomes the start face.

A.3.2 Agent. All MPN agents employ a network architecture able to exploit the graph structure of the data, consisting of two layers passing messages along the edges of the graph. The vertex features in the face-adjacency graph are as follows:

- **Points:** A 10×10 grid of 3D points sampled from the UV coordinate space of the B-Rep face and normalized to the bounding box of the target geometry.
- **Normals:** A 10×10 grid of 3D normal vectors sampled from the UV coordinate space of the B-Rep face.
- **Trimming Mask:** A 10×10 grid of binary values representing samples that are inside/outside the B-Rep face trimming boundary.
- **Surface Type:** A one-hot encoded flag indicating the type of surface represented by the B-Rep face: Cone, Cylinder, Elliptical, EllipticalCylinder, Nurbs, Plane, Sphere, Torus.

Using the face extrusion sequence data, we train the agents in an offline manner without interacting with the *Fusion 360 Gym*. The **mlp** and **gen** agents have a hidden dimension of 256 across all layers. The **gin** agent has two 256-dimensional linear layers within its graph convolution layer. The **gat** has 8 heads of 64 hidden dimensions each. The agents are trained with a dropout rate of 0.1 and a learning rate of 0.0001 for 100 epochs with the model saved at the lowest training loss. The learning rate is decreased by a

factor of 0.1 upon plateau within 10 most recent epochs. Training is performed on an NVIDIA Tesla V100 with an Adam optimizer and takes approximately 6-8 hours.

A.3.3 Search. In all search algorithms we mask out the following invalid actions so they are never taken:

- Start faces surface types that are non-planar
- End faces surface types that are non-planar
- Operation types other than *new body* when the current geometry is empty

Other invalid actions that require geometric checks, such as specifying a start face and end face that are co-planar, are returned as invalid from the *Fusion 360 Gym* and count against the search budget.

A.3.4 Evaluation. We perform evaluation using the official test set containing 1725 designs. Evaluation is performed in an online manner using the *Fusion 360 Gym*. Figure 31 shows the average reconstruction time for each design with combinations of agents and search strategies. We set a hard time limit of 10 minutes per design, after which we halt search, affecting between 0-14 designs depending on the agent and search strategy. Between 0-15 designs cause software crashes. 17 designs in the test set cannot be represented as graphs due to our data pipeline currently not supporting edges with more than two adjacent faces. In all failure cases we use

the best seen IoU, or 0 if no IoU score is available, and consider the design to fail at exact reconstruction.

A.3.5 Results. Table 6 details the full set of results for all agents and search strategies in the extended baseline comparison experiment from Section 6.1. We also include the number of parameters used by each agent.

A.4 Tasks

The ground-truth sequence of the dataset, along with the gym environment, can be used to automatically derive a range of labels for tasks other than CAD reconstruction, such as, program synthesis, sequence modeling, generative models, and geometric deep learning. Example tasks include:

- *Classification* of designs by construction sequence length.
- *Segmentation* of B-Rep faces by extrude operation order or by start/side/end face of an extrude operation.
- *Modeling operation order prediction* to recover the correct order of construction from raw geometry.
- *Sketch synthesis* to recover the original sketch, including constraints and dimensions, from the 3D geometry.
- *Predicting next action* in the design sequence for ‘CAD autocomplete’.
- *Generative models* that are aware of the design sequence and constraints.

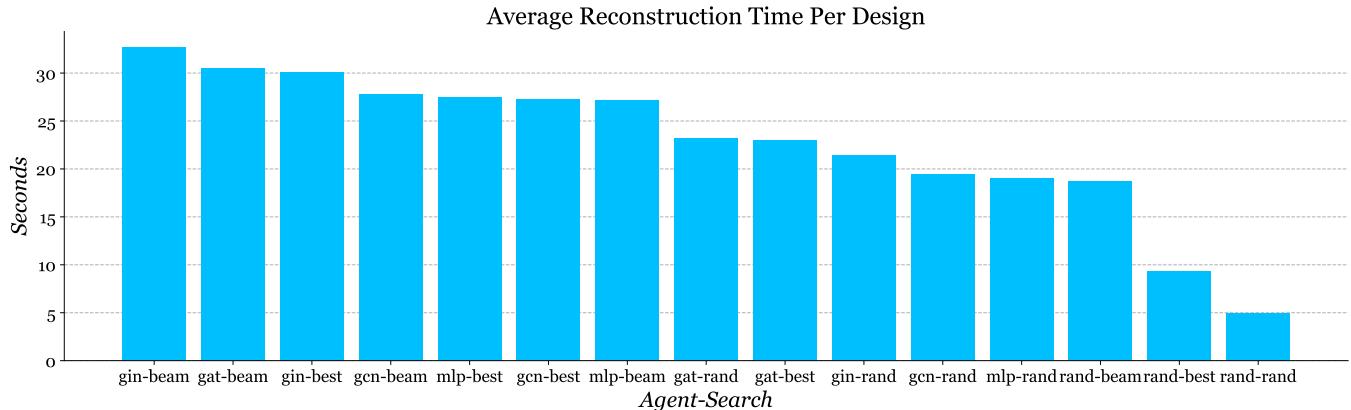


Fig. 31. Average reconstruction time per design for combinations of agents and search strategies.

Table 6. Reconstruction results for multiple agent and search combinations trained on human designed data. IoU and exact reconstruction are shown at 20 and 100 search steps. The best result in each column is shown in bold. Lower values are better for conciseness. # Parameters indicated the number of parameters used by each agent.

Agent	Search	IoU		Exact Reconstruction. %		Conciseness	# Parameters.
		20 Steps	100 Steps	20 Steps	100 Steps		
gcn	rand	0.8644	0.9042	0.6232	0.6754	1.0168	3.02M
gcn	beam	0.8640	0.8982	0.5739	0.6122	0.9275	3.02M
gcn	best	0.8831	0.9186	0.5971	0.6348	0.9215	3.02M
mlp	rand	0.8274	0.8596	0.5658	0.5965	0.9763	2.24M
mlp	beam	0.8619	0.8995	0.5525	0.5884	0.9271	2.24M
mlp	best	0.8712	0.8991	0.5675	0.5977	0.9305	2.24M
gat	rand	0.8742	0.9128	0.6191	0.6742	1.0206	3.03M
gat	beam	0.8691	0.9016	0.5791	0.6133	0.9261	3.03M
gat	best	0.8895	0.9139	0.5994	0.6354	0.9290	3.03M
gin	rand	0.8346	0.8761	0.5901	0.6301	1.0042	3.62M
gin	beam	0.8500	0.8913	0.5594	0.5983	0.9299	3.62M
gin	best	0.8693	0.9007	0.5803	0.6122	0.9340	3.62M
rand	rand	0.6840	0.8386	0.4157	0.5380	1.2824	-
rand	beam	0.4785	0.6277	0.2812	0.3896	0.9118	-
rand	best	0.6334	0.7994	0.3693	0.4887	0.8979	-